REPORT CSG-13

DECEMBER 1982

CSL COORDINATED SCIENCE LABORATORY

COMPUTER SYSTEMS GROUP

A MODEL FOR SIMULATING PHYSICAL FAILURES IN MOS VLSI CIRCUITS

PRITHVIRAJ BANERJEE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT CSG-13

DECEMBER 1982

A MODEL FOR SIMULATING PHYSICAL FAILURES IN MOS VLSI CIRCUITS

PRITHVIRAJ BANERJEE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A MODEL FOR SIMULATING PHYSICAL FAILURES IN MOS VLSI CIRCUITS		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER CSG-13
7. AUTHOR(s) Prithviraj Banerjee		8. CONTRACT OR GRANT NUMBER(s) N00039-80-C-0556
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Electronics Systems Command VHSIC Program		12. REPORT DATE December 1982 13. NUMBER OF PAGES 149
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Fault models
MOS VLSI circuits
circuit-level simulations
multi-valued algebra
physical failures
functional fault models
logic simulator

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Studies of the effect of physical failures on MOS logic circuits at the circuit level reveal the existence of failure modes that are not covered by existing fault models. Functional fault models for certain modules are discussed with a view towards designs for testability. Unfortunately, such detailed studies are not practical for complex VLSI modules. To analyze the logical effects of physical failures in MOS circuits, a multi-valued algebra has been constructed. The algebra has been verified by circuit level simulations for some typical VLSI modules such as decoders, programmable logic

DD FORM 1 JAN 73 1473

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

arrays and delay registers. This algebra can be used to study MOS circuits (both under failure and with no failure) with an accuracy much greater than that possible using gate-level simulators, and at speeds much greater than those possible using circuit level simulations. Contrary to other approaches of using an algebra to describe MOS circuit behavior, this is the first instance of an algebra trying to model physical failures. A simulator based on the algebra is also described in this thesis.

A MODEL FOR SIMULATING PHYSICAL FAILURES IN MOS VLSI CIRCUITS

BY

PRITHVIRAJ BANERJEE

B.Tech., Indian Institute of Technology, Kharagpur, 1981

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1983

Urbana, Illinois

ACKNOWLEDGMENT

The author wishes to express his deepest and heartfelt gratitude to his advisor Professor Jacob A. Abraham. Professor Abraham's constant patience, genuine personal warmth and insightful guidance were necessary ingredients to this research.

The author also wishes to thank Professors Edward S. Davidson and Janak Patel for their moral support and encouragement.

Finally, the author is deeply indebted to all his colleagues at the Computer Systems Group of the Coordinated Science Laboratory for their friendship and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Description of the Problem

The challenge of testing silicon integrated circuits (ICs) is becoming more formidable with the rapidly expanding production of very-large-scale integrated (VLSI) circuits. Increased gate count, pin limitations, smaller feature size, higher performance and higher complexity all contribute to an increasing testability problem.

As a further challenge, IC tests must be specifically designed to recognize failure-mode dependence upon circuit configuration, processing parameters and overall technology (TTL, ECL, NMOS, CMOS, etc.) [10,12,16]. That is, a Boolean network realized in one technology can have a strikingly different implementation in another. In addition, the types of physical failures which can occur vary greatly with the technology. Consequently, logic tests must be created which not only exercise the gross functional behaviour of the IC but also the structure used for that function. However, for VLSI circuits, internal circuit structure and complexity are increasing at a much more rapid rate than is the number of access terminals.

The increasing use of MOS technology has introduced a number of circuit elements whose logical behavior and faults cannot be described at the Boolean gate level. Recently, there has been some work done on building logic simulators that take care of some of the typical failure effects. Some of the circuit elements that were not covered by

conventional gate level simulators include transmission gates, tri-state inverters and bi-directional buses. Furthermore, the failure modes of such circuits can cause non-classical logic faults. That is, they possess a faulty behavior for which test coverage cannot be verified by a conventional logic simulator [8].

An accurate understanding of the effects of physical failures on digital systems is essential in order to design tests for them, and to design circuitry to detect or tolerate them. Such an understanding may be obtained by studying their effects at the circuit level. Unfortunately, such detailed studies are not practical for complex VLSI modules. It is therefore desirable to have a model for describing the effects of physical failures at the logical level.

The purpose of the research described in this thesis was to investigate and model the types of faults that might result from physical failures in NMOS and CMOS circuits which are two of the dominant technologies in VLSI today. Information regarding typical physical failure mechanisms was obtained from the field.

1.2. Background

Classical approaches to test generation for faults in logic circuits have assumed that physical failures can be modeled as lines in the gate-level description of the circuit "stuck" at "0" or "1". These logical faults were considered to occur at the inputs and outputs of logic gates.

A study on bridging faults [20] on logic gates revealed that certain classes of bridging faults (that give rise to inverting or non-inverting feedbacks) could be detected by a test set that is designed to detect stuck-at faults.

Wadsack [28,29] developed a fault model for describing faults in CMOS circuits. For CMOS circuits, it was reported that in addition to the logic levels "0" and "1", there exists a third logic state. This third condition is the "open" or high impedance state. One source of the "open" state is the presence of a fault which prevents one network from conducting when the other is in a non-conducting state. A second cause is the legitimate use of a high impedance state in dynamic circuits or tri-state buffers. In each instance the output retains the logic value of the previous output state. This is true because the gates are loaded with capacitance only. The length of time the state is retained is determined by the leakage current at the node. Such faults were modeled by the "stuck open" fault model.

A "stuck closed" model has been proposed to model the behavior of some networks remaining in the conducting state permanently.

1.3. Thesis Outline

In Chapters 2 and 3, some results on the effects of physical failures on certain basic cells, namely inverters, NAND gates, NOR gates, and latches, will be described for NMOS and CMOS circuits. The results were obtained by simulating the physical failures at the circuit level using the SPICE circuit simulation program [21]. In addition to

simulating "dead shorts" and "dead opens" different kinds of "resistive" shorts were also studied. Various interconnect failures such as opens in metal and polysilicon lines, shorts between diffusion lines were also considered. The results showed that some of the failures cannot be modeled by any of the existing fault models [2].

In Chapter 4, functional fault models for some typical NMOS modules are presented. These models were obtained from a detailed circuit level study of physical failures. Obviously, the question arises as to whether it is worthwhile to approach the problem from the circuit level. In view of the immense complexity of VLSI circuits today, efforts have been directed towards obtaining functional fault models which reduce the complexity of test generation. However the fault models considered in many of these approaches are based on assumptions that have relatively little correspondence with the failures that occur in the field. A study of the effects of physical failures at the circuit level may be used to come up with accurate fault models.

A detailed study of physical failures at the circuit level is obviously impossible for complex modules. A two-step approach was therefore used, whereby detailed studies were performed on relatively small blocks and these results were then used to analyze the behavior of complex VLSI circuits. After extensive circuit simulations of faulty logic devices, approximate models for MOS transistors were developed; a multi-valued logic algebra was constructed with the necessary set of rules to describe MOS circuit behavior. This will be discussed in Chapter 5.

This work has resulted in a new MOS logic simulator called MURPHY (which is an acronym for Mos logic simulation Under Realistic PHYsical failures). It can be used to model the behavior of VLSI circuits (both under failure and with no failure) with an accuracy much greater than that using gate level simulators, and at speeds much greater than those possible using circuit simulators. The fault models used in the simulator are derived directly from the physical failures. The simulator is about two orders of magnitude faster than SPICE. A brief description of the simulator is presented in Chapter 6.

CHAPTER 2

EFFECTS OF PHYSICAL FAILURES IN BASIC NMOS CELLS

2.1. Introduction

Before describing the results of the study of the effects of physical failures on basic NMOS cells, a brief description of some of the NMOS physical failure modes, typically observed in the field [31], will be presented. The failure modes are broadly classified into three groups, depending on their frequency of occurrence. Class I failures are the most likely, Class II failures are moderately likely, and Class III failures are the least likely to occur.

Among each group there are two subgroups: (1) Device failures, and (2)Interconnect failures.

Device failures refer to failures within an active device, namely an NMOS transistor. Failures may include shorts between the gate and source of a transistor, floating gates, and opens in the source and drain lines.

Interconnect failures refer to failures in the lines that connect the active devices. These include opens in metal and polysilicon lines, shorts between diffusion lines and so on.

Some of the failures are listed below:

Class I (most likely):

Device:

Gate to Drain short.

Gate to Source short.

Interconnect:

Short between Diffusion lines.

Class II (moderately likely):

Device:

Drain contact open.

Source contact open.

Interconnect:

Aluminium polysilicon cross-over broken.

Class III (less likely):

Device:

Gate to Substrate short.

Floating gate.

Interconnect:

Short between Aluminium lines.

In this chapter, we shall discuss the effects of the above physical failures on certain primitive NMOS cells such as inverters, NAND gates, NOR gates, and dynamic latches. Each of these circuits was simulated under fault-free and faulty conditions, and their results were compared. The effects of gradual degradation were also studied. For example, when a short between the gate and drain of a transistor was simulated, the effect of gradually varying the resistance between the gate and drain on the behavior was also studied.

2.2. NMOS Inverter

Figure 2.1 shows some of the physical failures possible in an NMOS inverter. Each of these failures will be individually considered. The effect of gradual deterioration was considered by performing the SPICE simulation for various values of resistance at the location of the failure.

Fault 1: Short between drain and gate of enhancement transistor

From Table 2.1, it is observed that for resistance values above 2000 Kilo-ohms (abbreviated K-ohms), there is no error at all. As the resistance value decreases to about 100 K-ohms, the high input to the faulty inverter becomes about 3.3 V, and the corresponding output becomes 1.1 V. The normally low input remains low around 0.5 V.

However, in the resistance range around 50 K-ohms, the output is stuck at a voltage level around 1.9 V, which can be considered to be a new logic level, "0*"; this level has a different logical behavior from a logic "0". Specifically, if such a voltage level is applied to the gate of a pass transistor in a dynamic latch, it can discharge any charge previously stored in the latch. But it is recognized as a "0" by a logic gate.

For resistance values around 30 K-ohms, the output voltage level corresponds to an indeterminate logic level, which will be labeled as "I". Such a level has the property that it may be recognized as either a "0" or a "1" by a logic gate, depending on its logic threshold.

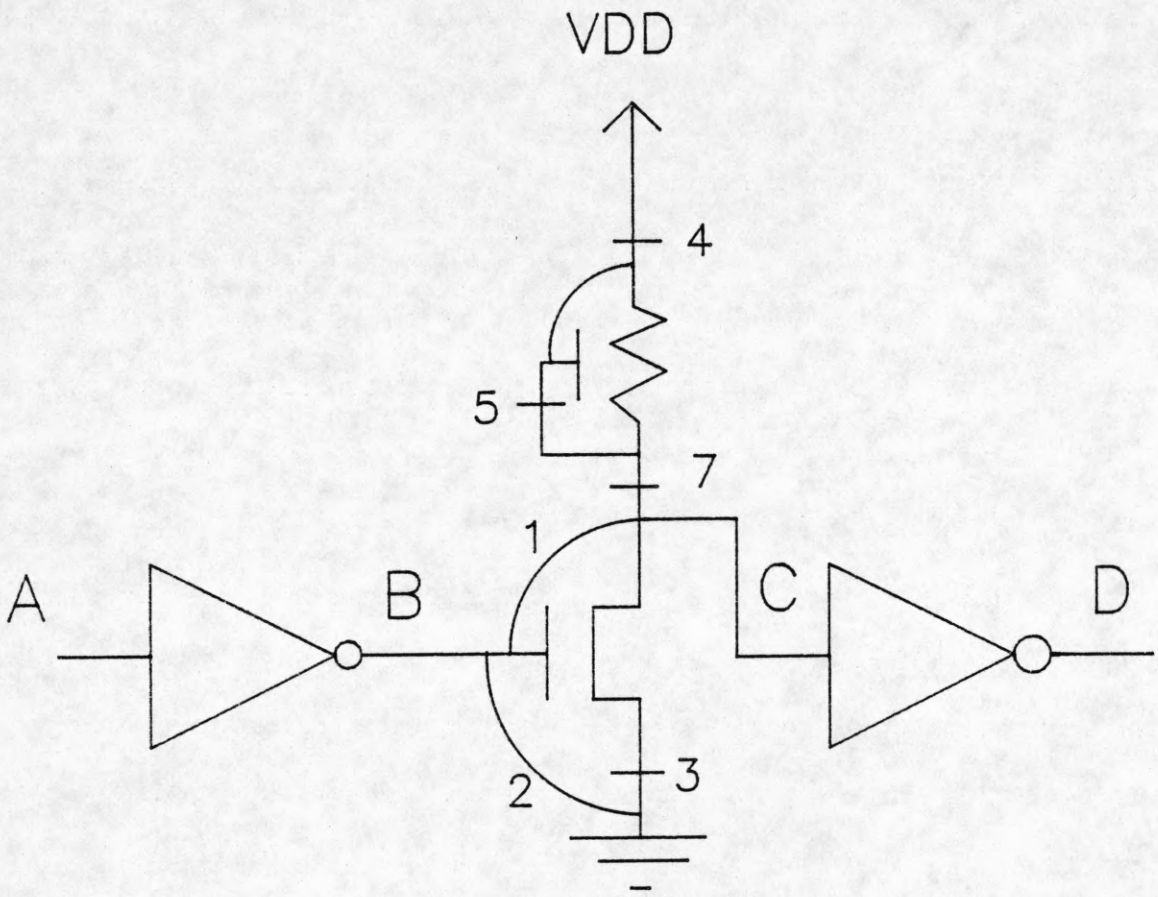


Figure 2.1: NMOS Inverter

Table 2.1: Fault 1 in NMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
2000	0.00	5.00	0.29	5.00	Fault-free output
	5.00	0.29	5.00	0.29	
100	0.00	3.30	1.15	4.40	No error
	5.00	0.50	3.10	0.40	
50	0.00	3.06	1.87	4.44	Stuck-at-0
	5.00	0.60	1.97	4.50	
30	0.00	3.04	2.33	3.38	Indeterminate
	5.00	0.60	1.45	4.48	
10	0.00	3.02	2.78	1.00	Error
	5.00	0.60	0.88	4.70	
0	0.00	3.00	3.00	0.80	Error
	5.00	0.60	0.60	4.95	

The most interesting observations are for the values of resistance less than 10 K-ohms. Here, the output of the inverter follows the input, instead of being inverted. The normally high input (5.0 V) becomes 3.0 V, and the output becomes 3.0 V, instead of 0.3 V. The normally low input (0.3 V) remains low (0.6 V), but the output becomes 0.6 V, instead of 5.0 V. Hence both the input and the output of the faulty inverter are affected. The voltage level (3.0 V) represents logic "1*". Such a level is recognized as a logic "1" by a logic gate; but it fails to turn a pass transistor fully on. A pass transistor having a level "1*" at its gate can pull the drain low, but cannot pull it fully high.

Fault 2: Short between source and gate of enhancement transistor

From Table 2.2, it is noted that for high resistance shorts, there is no error. However, as the resistance drops to about 95 K-ohms, the gate exhibits a faulty behavior. The value of the resistance at the transition point is extremely critical. For resistance values above 95 K-ohms, there is no error; but for resistance values around 93 K-ohms, the circuit outputs a voltage level which corresponds to logic "I". For resistance values around 70 K-ohms and below, the inverter output is stuck at "1".

Fault 3: Source contact of enhancement transistor open

From Table 2.3, it can be seen that for resistance values above 100 K-ohms, the inverter output is stuck at "1". It is to be noted, however, that this fault is much less likely to occur than faults 1 and 2, because this failure happens to be a Class II failure.

Table 2.2: Fault 2 in NMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
2000	0.00	5.00	0.29	5.00	Fault-free output
	5.00	0.29	5.00	0.29	
100	0.00	2.03	1.03	4.95	No error
	5.00	0.50	4.60	0.38	
95	0.00	2.51	1.35	4.92	No error
	5.00	0.26	4.99	0.26	
93	0.00	2.47	2.62	1.01	Indeterminate
	5.00	0.26	4.95	0.26	
70	0.00	1.90	4.39	0.39	Stuck-at-1
	5.00	0.25	4.99	0.28	
1	0.00	0.50	5.00	0.26	Stuck-at-1
	5.00	0.20	5.00	0.26	

Table 2.3: Fault 3 in NMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
1	0.00	5.00	0.45	5.00	Fault-free output
	5.00	0.29	5.00	0.26	
10	0.00	5.00	0.58	5.00	No error
	5.00	0.29	5.00	0.25	
100	0.00	5.00	3.11	0.63	Stuck-at-1
	5.00	0.30	4.98	0.29	
1000	0.00	5.00	4.80	0.35	Stuck-at-1
	5.00	0.29	4.99	0.29	

Fault 4: Drain contact of depletion transistor open

Referring to Table 2.4, it is observed that an open contact in the drain of the depletion transistor results in a stuck-at-0 fault at the output of the inverter, a result which is entirely expected since the power to the inverter is removed.

Fault 5: Floating gate of depletion transistor

The SPICE program does not allow floating nodes due to convergence problems. Hence this fault was simulated by removing the connection from gate to source of the depletion transistor and attaching a resistance from the gate to the substrate. By varying the resistance value, two physical failure mechanisms were simulated, namely, a floating gate, and a short between the gate and the substrate. From Table 2.5, it is seen that the floating gate (simulated by connecting a 1000 K-ohms resistance between the gate and the substrate) results in no error at all. This is interesting because one would expect some effect of the failure at the logic level.

For the case of a short between the gate and substrate, the same indeterminate error was observed. Instead of a voltage level of 5.0 V for a logic "1", the output became 3.0 V. The logic "0" was not affected.

Fault 6: Short between drain and gate of depletion transistor

For this fault, the output remained stuck at "1".

Table 2.4: Fault 4 in NMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
1	0.00	5.00	0.45	5.00	Fault-free output
	5.00	0.29	5.00	0.26	
1000	0.00	5.00	0.10	5.00	Stuck-at-0
	5.00	0.26	0.60	5.00	

Table 2.5: Fault 5 in NMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
1000	0.00	5.00	0.24	5.00	No error
	5.00	0.29	4.63	0.32	
10	0.00	5.00	0.24	5.00	Some error
	5.00	0.29	2.98	0.68	
5	0.00	5.00	0.25	5.00	Some error
	5.00	0.29	2.98	0.68	
0	0.00	5.00	0.25	5.00	Some error
	5.00	0.29	2.98	0.68	

Fault 7: Source contact of depletion transistor open

For this fault, the output remained stuck at "0".

Fault 8: Gate to source connection of depletion transistor resistive

This failure is an extension of the failure mentioned under fault 5. A resistance is placed between the gate and source of the depletion transistor. Again there is no effect of the failure on the logical behavior of the inverter (Table 2.6).

Effect of geometry of transistors on fault 1

The effect of different geometries of the transistors on a short between the gate and drain of the enhancement transistor was studied. The results are shown in Table 2.7. The results show that the voltage corresponding to the logic level "1*" varies by about 0.5 V.

2.3. NMOS NAND Gate

The effects of physical failures on NMOS NAND gates will now be discussed. Figure 2.2 shows a two input NAND gate under various failures. In order to simulate the effects of the failures, it was necessary to drive the NAND gate through buffers.

Fault 1: Short between drain and gate of lower enhancement transistor

Referring to Table 2.8, it can be seen that error-free behavior is obtained for values of resistance greater than 2000 K-ohms. As the resistance value decreases to about 100 K-ohms, the first indication of an error occurs. For the input combination, A = "0", B = "1", the output

Table 2.6: Fault 8 in NMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
0	0.00	5.00	0.26	5.00	Fault-free output
	5.00	0.26	5.00	0.26	
1000K	0.00	5.00	0.29	5.00	No error
	5.00	0.29	5.00	0.29	

Table 2.7: Effect of Geometry of Transistors on Fault 1 in NMOS Inverter

Device dimensions (Length L, Width W in microns)								Simulation results		
m1		m2		m3		m4		V(A)	V(B)	V(D)
L	W	L	W	L	W	L	W	(volt)	(volt)	(volt)
4	4	16	4	4	4	16	4	0.00	2.90	0.70
								5.00	0.60	5.00
4	4	16	4	4	8	8	4	0.00	2.75	0.85
								5.00	1.01	5.00
4	4	16	4	4	16	4	4	0.00	2.61	1.02
								5.00	1.65	4.70
4	4	16	4	4	32	4	8	0.00	2.54	1.20
								5.00	2.13	3.97
4	8	8	4	4	4	16	4	0.00	3.29	0.56
								5.00	0.45	5.00
4	16	4	4	4	4	16	4	0.00	2.98	0.69
								5.00	0.17	5.00
4	16	4	4	4	16	4	4	0.00	2.62	1.02
								5.00	0.47	5.00
4	8	8	4	4	8	8	4	0.00	2.96	0.70
								5.00	0.66	5.00

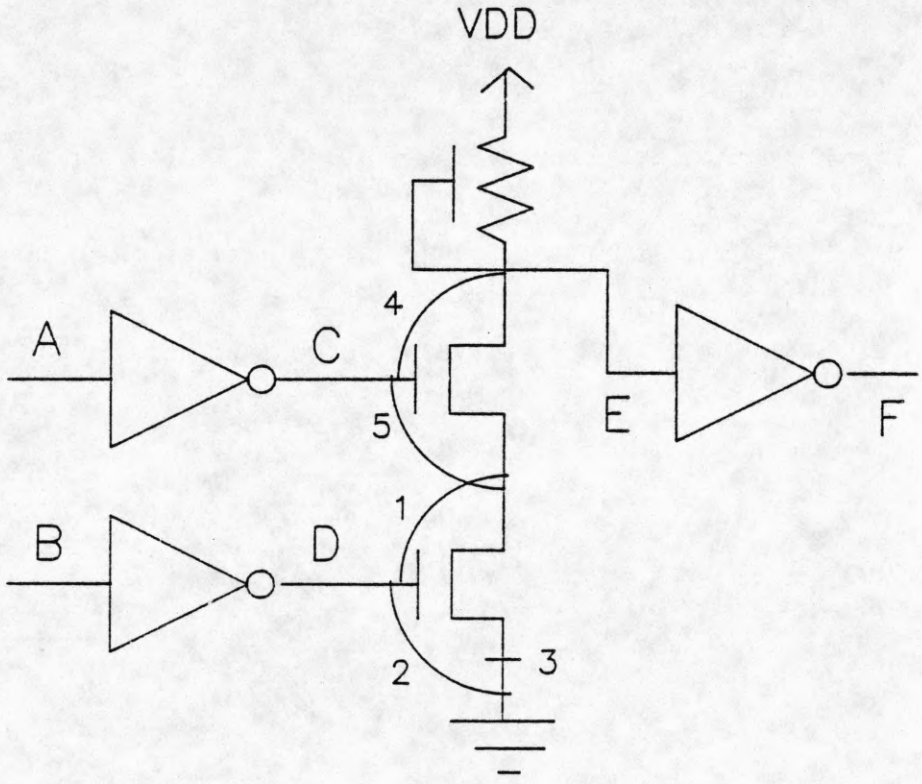


Figure 2.2: NMOS NAND Gate

Table 2.8: Fault 1 in NMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
2000	5.00	5.00	5.00	0.30	Fault free o/p.
	5.00	0.30	5.00	0.30	
	0.30	0.30	0.30	5.00	
	0.30	5.00	5.00	0.30	
100	5.00	5.00	5.00	0.29	Slight error.
	5.00	0.30	5.00	0.29	
	0.30	0.30	0.30	5.00	
	0.30	5.00	2.90	0.75	
40	5.00	5.00	5.00	0.30	Type 1 error.
	5.00	0.30	5.00	0.30	
	0.30	0.30	1.66	4.61	
	0.30	5.00	1.93	4.39	
10	5.00	5.00	5.00	0.30	Type 2 error.
	5.00	0.30	5.00	0.30	
	0.30	0.30	2.58	1.11	
	0.30	5.00	1.11	5.00	
0	5.00	5.00	5.00	0.30	Type 2 error.
	5.00	0.30	5.00	0.30	
	0.30	0.30	2.90	0.72	
	0.30	5.00	0.88	4.97	

becomes 2.9 V instead of 5.0 V. When the resistance becomes 40 K-ohms, the fault can be modeled as an input stuck at "1". For even lower values of the resistance, i.e., 10 K-ohms or less, a totally different logical behavior is noted. For the input combination, A = "0", B = "0", the output becomes 2.6 V instead of 0.3 V, whereas for the input combination, A = "0", B = "1", the output is 1.1 V instead of 5.0 V. This faulty behavior cannot be modeled as a stuck-at fault. The voltage level 2.6 V corresponds to the indeterminate logic level "I".

Fault 2: Short between source and gate of lower enhancement transistor

This failure can be modeled as the output stuck at "1" for low values of the resistance between the source and the gate of the faulty transistor (refer Table 2.9). For resistance values around 70 K-ohms, the output is 3.5 V instead of 5.0 V for the input combination, A = "0", B = "0". This voltage level corresponds to the logic level "1*" mentioned earlier.

Fault 3: Source contact of lower enhancement transistor open

From Table 2.10, it is seen that the logical effect of this failure is identical to that of fault 2. For very high values of the resistance (which approximates an open in the source to ground line), the fault can be modeled as the output stuck at "1". For intermediate values of the resistance, around 70 to 100 K-ohms, the output becomes about 2.6 V for input combination, A = "0", B = "0", which corresponds to the indeterminate logic level "I".

Table 2.9: Fault 2 in NMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
2000	5.00	5.00	5.00	0.30	Fault free o/p.
	5.00	0.30	5.00	0.30	
	0.30	0.30	0.30	5.00	
	0.30	5.00	5.00	0.30	
100	5.00	5.00	5.00	0.30	No error.
	5.00	0.30	5.00	0.30	
	0.30	0.30	0.57	5.00	
	0.30	5.00	4.53	0.30	
70	5.00	5.00	5.00	0.30	Indeterminate.
	5.00	0.30	5.00	0.29	
	0.30	0.30	3.55	0.48	
	0.30	5.00	4.97	0.30	
40	5.00	5.00	5.00	0.30	Stuck-at-1.
	5.00	0.30	5.00	0.30	
	0.30	0.30	4.95	0.30	
	0.30	5.00	4.99	0.30	

Table 2.10: Fault 3 in NMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
0	5.00	5.00	5.00	0.30	Fault free o/p.
	5.00	0.30	5.00	0.30	
	0.30	0.30	0.30	5.00	
	0.30	5.00	5.00	0.30	
70	5.00	5.00	5.00	0.30	Indeterminate error.
	5.00	0.30	5.00	0.30	
	0.30	0.30	2.56	1.15	
	0.30	5.00	4.80	0.32	
100	5.00	5.00	5.00	0.30	Indeterminate error.
	5.00	0.30	5.00	0.30	
	0.30	0.30	3.13	0.60	
	0.30	5.00	4.90	0.30	
1000	5.00	5.00	5.00	0.30	Stuck-at-1.
	5.00	0.30	5.00	0.30	
	0.30	0.30	4.80	0.32	
	0.30	5.00	4.90	0.30	

Fault 4: Short between drain and gate of upper enhancement transistor

Referring to Table 2.11, it is observed that for values of the resistance greater than 2000 K-ohms, there is no error. As the resistance value is decreased to about 100 K-ohms, the first type of error is observed. For input combinations A = "1", B = "1" and A = "1", B = "0", the output is 3.0 V. This corresponds to the logic level "1*". As the resistance drops to about 40 K-ohms, a second type of error is observed. The output is at logic "0*" for all input combinations except for A = "0", B = "1".

For resistance values below 10 K-ohms, a third kind of error is seen. The output becomes 0.6 V instead of 5.0 V for two input combinations. For the input combination A = "0", B = "1", the output remains at 5.0 V, but for A = "0", B = "0", the output becomes 2.7 V instead of 0.3 V.

Fault 5: Short between gate and source of upper enhancement transistor

From Table 2.12, it is noted that there is no error as long as the resistance value remains more than 100 K-ohms. For resistance values below 40 K-ohms, the output is stuck at "1". However, for intermediate values of the resistance around 70 K-ohms, the output becomes 3.6 V for input combination A = "0", B = "0".

2.4. NMOS NOR Gate

The next basic cell to be considered is a two input NOR gate. Failures in the depletion transistor will not be considered because their effects are very similar to those on the inverter, considered

Table 2.11: Fault 4 in NMOS NAND gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
2000	5.00	5.00	5.00	0.30	Fault free o/p.
	5.00	0.30	5.00	0.30	
	0.30	0.30	0.30	5.00	
	0.30	5.00	5.00	0.30	
100	5.00	5.00	2.99	0.68	Type 1 error.
	5.00	0.30	2.99	0.68	
	0.30	0.30	0.92	5.00	
	0.30	5.00	4.98	0.30	
40	5.00	5.00	1.69	4.66	Type 2 error.
	5.00	0.30	1.69	4.66	
	0.30	0.30	1.74	4.61	
	0.30	5.00	4.98	0.30	
0	5.00	5.00	0.60	5.00	Type 3 error.
	5.00	0.30	0.60	5.00	
	0.30	0.30	2.73	0.60	
	0.30	5.00	5.00	0.30	

Table 2.12: Fault 5 in NMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
100	5.00	5.00	5.00	0.30	Fault-free o/p.
	5.00	0.30	5.00	0.30	
	0.30	0.30	0.71	5.00	
	0.30	5.00	4.96	0.30	
70	5.00	5.00	5.00	0.30	
	5.00	0.30	5.00	0.30	
	0.30	0.30	3.60	0.47	
	0.30	5.00	5.00	0.30	
40	5.00	5.00	5.00	0.30	Stuck-at-1
	5.00	0.30	4.99	0.30	
	0.30	0.30	5.00	0.30	
	0.30	5.00	5.00	0.30	

earlier. Also, since a NOR gate is symmetric, in the sense, that the effect of a the failure on any enhancement transistor is identical to that on any other, there are only two failures of interest for a NOR gate. Figure 2.3 shows the locations of these failures.

Fault 1: Short between drain and gate of enhancement transistor

The results of SPICE simulation for this failure are shown in Table 2.13. For resistance values around 2000 K-ohms, no error is observed. However, as the resistance drops to about 100 K-ohms, a slight error is observed for the input combination, A = "1", E = "1", when the output becomes 3.0 V instead of 5.0 V. This voltage level corresponds to the logic level "1*" mentioned earlier.

As the resistance drops to about 40 K-ohms, the output becomes stuck at "0*". For even lower values of the resistance, the behavior is totally different. Both the input B and the output C are altered. For input combination, A = "1", E = "1", the output becomes 0.6 V instead of 5.0 V, and B remains unaffected.

This fault has very special consequences as will be shown later due to the feedback from the output to the input.

Fault 2: Short between source and gate of enhancement transistor

Referring to Table 2.14, it is seen that this simply corresponds to a stuck-at-0 fault of the corresponding input of the NOR gate.

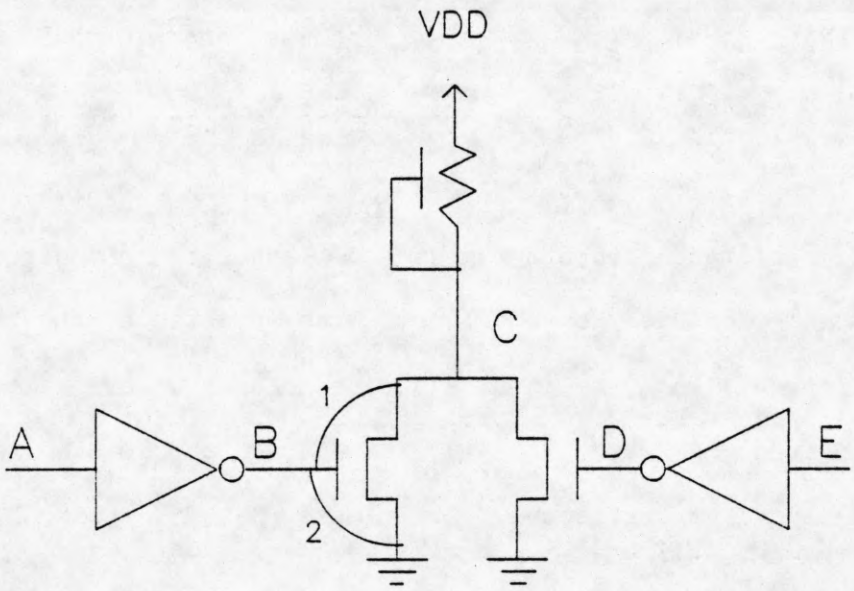


Figure 2.3: NMOS NOR Gate

Table 2.13: Fault 1 in NMOS NOR Gate

R (K-ohm)	V(A) (volt)	V(E) (volt)	V(D) (volt)	V(B) (volt)	V(C) (volt)	COMMENTS
2000	5.00	5.00	0.30	0.30	5.00	Fault free o/p.
	0.30	5.00	0.30	5.00	0.30	
	0.30	0.30	5.00	5.00	0.30	
	5.00	0.30	5.00	0.30	0.30	
100	5.00	5.00	0.30	0.52	3.00	Slight error.
	0.30	5.00	0.30	3.34	1.20	
	0.30	0.30	4.98	2.88	0.50	
	5.00	0.30	5.00	0.29	0.30	
40	5.00	5.00	0.30	0.61	1.70	Indeterminate.
	0.30	5.00	0.29	3.05	2.10	
	0.30	0.30	4.97	1.65	0.56	
	5.00	0.30	5.00	0.30	0.30	
1	5.00	5.00	0.29	0.61	0.61	Error.
	0.30	5.00	0.30	3.02	3.00	
	0.30	0.30	4.97	0.64	0.64	
	5.00	0.30	5.00	0.29	0.29	

Table 2.14: Fault 2 in NMOS NOR Gate

R (K-ohm)	V(A) (volt)	V(E) (volt)	V(D) (volt)	V(B) (volt)	V(C) (volt)	COMMENTS
1	5.00	5.00	0.30	0.26	5.00	Error.
	0.30	5.00	0.30	0.26	5.00	
	0.30	0.30	5.00	0.30	0.30	
	5.00	0.30	5.00	0.30	0.30	

2.5. NMOS Dynamic Latch

The effects of physical failures on an NMOS dynamic latch will be described next. The circuit diagram of a dynamic latch indicating the different failure locations is shown in Figure 2.4.

Fault 1: Short between source and gate of pass transistor

Referring to Table 2.15, it is observed that this failure has interesting effects. In NMOS technology, the joining of the outputs of several logic gates produces a wired-AND operation. Since the source and gate of the pass transistor are shorted, the resultant signal driving the source and gate of the pass transistor is obtained by AND-ing the outputs of the two inverters driving them. The gate can be high only when both the gate and the source are high. It is therefore only possible to load a "1*" and not a "0" into the faulty latch. Hence F becomes stuck at "1*" for some time after a "1*" is loaded. However, if the gate is not refreshed often enough, the charge on the latch will leak away. It is therefore not a stuck-at fault in the true sense.

Fault 2: Short between drain and gate of pass transistor

The simulation results for this failure are shown in Table 2.16. For resistance values in the range of about 2000 K-ohms, there is no error for short periods of time. The "lifetime" of the stored logic "1" at the gate of the latch is quite small - the exact value depending on the resistance value. It may be noted that the charge will leak away if not refreshed very frequently. This behavior continues till the resistance value falls to about 200 K-ohms.

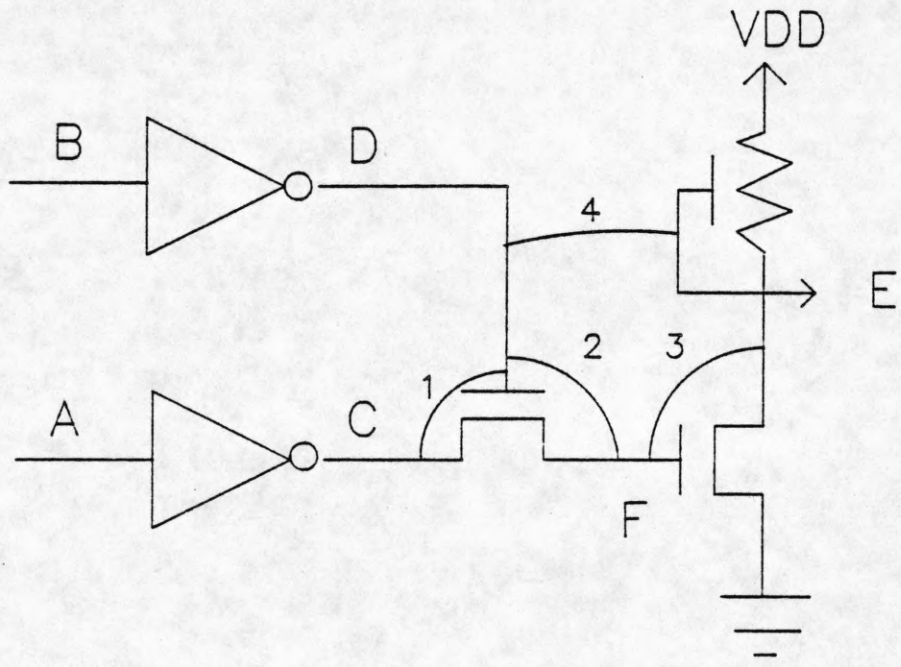


Figure 2.4: NMOS Dynamic Latch

Table 2.15: Fault 1 in NMOS Dynamic Latch

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(D) (volt)	V(C) (volt)	V(F) (volt)	V(E) (volt)	COMMENTS
2000	0.30	5.00	0.30	5.00	0.00	5.00	Fault free o/p.
	0.30	0.30	5.00	5.00	3.60	0.36	One loaded.
	0.30	5.00	0.30	5.00	3.60	0.36	Data retained.
	5.00	5.00	0.30	0.30	3.60	0.36	
	5.00	0.30	5.00	0.30	0.30	5.00	Zero stored.
	5.00	5.00	0.30	0.30	0.30	5.00	Data retained.
100	0.30	5.00	0.57	0.70	0.00	5.00	No error.
	0.30	0.30	4.97	5.00	3.55	0.30	
	0.30	5.00	0.57	0.70	3.55	0.30	
	5.00	5.00	0.30	0.30	3.55	0.30	
	5.00	0.30	2.99	0.57	0.57	5.00	
	5.00	5.00	0.30	0.30	0.57	5.00	
40	0.30	5.00	0.61	1.70	0.00	5.00	F (initially 0)
	0.30	0.30	4.95	4.95	3.53	0.27	is driven to 1.
	0.30	5.00	0.61	1.70	3.06	0.30	It remains at
	5.00	5.00	0.30	0.30	3.06	0.30	1 for some time
	5.00	0.30	1.70	0.61	3.06	0.30	but can later
	5.00	5.00	0.30	0.30	3.00	0.30	discharge.
10	0.30	5.00	0.61	0.90	0.00	5.00	Same error.
	0.30	0.30	4.94	4.96	3.50	0.30	
	0.30	5.00	0.61	0.90	3.50	0.30	
	5.00	5.00	0.30	0.30	3.45	0.30	
	5.00	0.30	0.88	0.61	3.42	0.30	
	5.00	5.00	0.30	0.30	3.41	0.30	

Table 2.16: Fault 2 in NMOS Dynamic Latch

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(D) (volt)	V(C) (volt)	V(F) (volt)	V(E) (volt)	COMMENTS
90000	5.00	0.30	5.00	0.30	0.30	5.00	Fault free o/p.
	0.30	0.30	5.00	5.00	3.80	0.30	One loaded.
	0.30	5.00	0.30	5.00	3.80	0.30	Data retained.
	5.00	5.00	0.30	0.30	3.80	0.30	
	5.00	0.30	5.00	0.30	0.30	5.00	Zero loaded.
2000	5.00	0.30	5.00	0.30	0.30	5.00	Memory loss.
	0.30	0.30	5.00	5.00	3.80	0.30	
	0.30	5.00	0.30	5.00	0.30	5.00	
	5.00	5.00	0.30	0.30	0.30	5.00	
	5.00	0.30	5.00	0.30	0.30	5.00	
200	5.00	0.30	3.90	0.40	0.70	5.00	Memory loss.
	0.30	0.30	4.80	5.00	4.50	0.30	
	0.30	5.00	0.30	5.00	0.30	5.00	
	5.00	5.00	0.30	0.30	0.30	5.00	
	5.00	0.30	3.90	0.40	0.30	5.00	
50	5.00	0.30	3.00	0.60	1.80	4.00	F follows D.
	0.30	0.30	5.00	5.00	5.00	0.30	Sequential
	0.30	5.00	0.50	5.00	0.30	5.00	circuit becomes
	5.00	5.00	0.30	0.50	0.30	5.00	combinational.
	5.00	0.30	3.00	0.60	1.80	4.00	

For some intermediate range of resistance values around 50 K-ohms, an entirely different behavior is observed. The circuit becomes combinational in nature; the logical behavior can be seen from the table.

For even smaller values of the resistance less than 1K, this fault converts the sequential action of the dynamic latch into a pure combinational circuit. F follows D independent of C; hence E follows B after two inverter delays. This will have special significance in modules which require the input data to be latched in before the combinational logic inside starts computing the output, for example in a PLA. Then the input will be assumed to be "stuck- at-1" for the period during which the gating signal is high. Subsequently, when the gating signal goes down the input will appear to be "stuck-at-0".

Fault 3: Short between drain and gate of the inverter

Referring to Table 2.17, the following points should be noted. For most input combinations, the output behaves as though it is stuck at "0*". However, for the case when the drain of pass transistor tries to go high while the gate is high, the output reaches the logic level "I". Another point to be noted is that the sequential behavior of the latch is lost.

If the short is resistive (about 100 K-ohms), then the behavior is different. When the gate of the pass transistor is low, the output is stuck at "0*". When the gate is high the latch behaves almost correctly; that is, it outputs a "0" when the input is "1", but outputs a "1*"

Table 2.17: Fault 3 in NMOS Dynamic Latch

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(D) (volt)	V(C) (volt)	V(F) (volt)	V(E) (volt)	COMMENTS
2000	0.30	5.00	0.30	5.00	0.00	5.00	Fault free o/p.
	0.30	0.30	5.00	5.00	3.60	0.36	One loaded.
	0.30	5.00	0.30	5.00	3.60	0.36	Data retained.
	5.00	5.00	0.30	0.30	3.60	0.36	
	5.00	0.30	5.00	0.30	0.30	5.00	Zero loaded.
	5.00	5.00	0.30	0.30	0.30	5.00	Data retained.
200	0.30	5.00	0.30	5.00	2.06	2.06	Error.
	0.30	0.30	5.00	4.16	2.98	0.48	
	0.30	5.00	0.30	5.00	2.06	2.06	
	5.00	5.00	0.30	0.30	2.06	2.06	
	5.00	0.30	5.00	0.48	0.65	3.90	
	5.00	5.00	0.30	0.30	2.06	2.06	
100	0.30	5.00	0.30	5.00	2.06	2.06	Error.
	0.30	0.30	5.00	3.61	2.68	0.74	
	0.30	5.00	0.30	5.00	2.06	2.06	
	5.00	5.00	0.30	0.30	2.06	2.06	
	5.00	0.30	5.00	0.56	0.85	3.16	
	5.00	5.00	0.30	0.30	2.06	2.06	
40	0.30	5.00	0.30	5.00	2.06	2.06	Error.
	0.30	0.30	5.00	3.26	2.44	1.53	
	0.30	5.00	0.30	5.00	2.06	2.06	
	5.00	5.00	0.30	0.30	2.06	2.06	
	5.00	0.30	5.00	0.61	0.96	2.05	
	5.00	5.00	0.30	0.30	2.06	2.06	
0	0.30	5.00	0.30	5.00	2.06	2.06	Error.
	0.30	0.30	5.00	3.25	2.43	2.43	
	0.30	5.00	0.30	5.00	2.06	2.06	
	5.00	5.00	0.30	0.30	2.06	2.06	
	5.00	0.30	5.00	0.61	0.96	0.96	
	5.00	5.00	0.30	0.30	2.06	2.06	

(close to 1) when the input is "0".

Fault 4: Gate of pass transistor shorted to output

From Table 2.18, it is seen that for resistance values around 2000 K-ohms, no error is observed. As the resistance value is decreased to about 200 K-ohms, there is still no error in the output but the voltage value stored in the latch is about 2.8 V. Furthermore, the input, driving the gate of the faulty transistor, is pulled to 3.8 V when the source of the pass transistor is at 5.0 V. This behavior continues till the resistance is about 40 K-ohms.

For even lower resistance values, the behavior of the circuit tends to become combinational in nature. There are again three voltage levels at the output, namely logic "0", logic "I" and logic "1". At this stage D and E follow each other. However, this fault is different from Fault 2 in that, the latter shows only two logic levels at the output, whereas this fault produces three.

2.6. Conclusion

For each of the basic cells, the results shown were special in some respects. Some of the failures in the NAND and NOR gates that were similar to the failures in the inverter considered earlier, were not shown separately. These included failures such as a floating gate, or a short between the gate and substrate of the depletion transistor, or an open in the drain to VDD line.

It may be concluded from the above results that there exists a sizeable fraction of physical failures in NMOS circuits which do not

Table 2.18: Fault 4 in NMOS Dynamic Latch

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(D) (volt)	V(C) (volt)	V(F) (volt)	V(E) (volt)	COMMENTS
2000	5.00	0.30	5.00	0.30	0.30	5.00	Fault-free
	0.30	0.30	5.00	5.00	3.60	0.30	One loaded.
	0.30	5.00	0.30	5.00	3.60	0.30	Data retained.
	5.00	5.00	0.30	0.30	3.60	0.30	
	5.00	0.30	5.00	0.30	0.30	5.00	Zero retained.
200	5.00	0.30	5.00	0.30	0.30	5.00	No error in o/p
	0.30	0.30	3.80	5.00	2.80	0.60	but error in
	0.30	5.00	0.30	5.00	2.70	0.60	memory.
	5.00	5.00	0.30	0.30	2.70	0.50	
	5.00	0.30	5.00	0.30	0.30	5.00	
100	5.00	0.30	5.00	0.30	0.30	5.00	Similar error.
	0.30	0.30	3.20	5.00	2.60	0.90	
	0.30	5.00	0.50	5.00	2.40	0.90	
	5.00	5.00	0.50	0.30	2.40	0.90	
40	5.00	0.30	5.00	0.50	0.50	5.00	Similar error.
	0.30	0.30	2.20	5.00	2.50	1.20	
	0.50	5.00	0.40	5.00	2.30	0.50	
	5.00	5.00	0.40	0.50	2.30	0.50	
	5.00	0.30	5.00	0.50	0.50	5.00	
10	5.00	0.30	5.00	0.50	0.50	5.00	Error.
	0.30	0.30	2.50	5.00	2.50	2.20	
	0.30	5.00	0.50	5.00	2.20	0.50	
	5.00	5.00	0.50	0.50	2.20	0.50	
	5.00	0.30	5.00	0.50	0.50	5.00	
0	5.00	0.30	5.00	0.50	0.50	5.00	Error.
	0.30	0.30	2.80	5.00	2.40	2.80	
	0.30	5.00	0.40	5.00	2.10	0.40	
	5.00	5.00	0.40	0.40	2.10	0.40	
	5.00	0.30	5.00	0.50	0.50	5.00	

result in stuck-at faults. In addition to a "hard" zero (logic "0") and a "hard" one (logic "1"), there exist intermediate logic levels ("0*","I", and "1*"), which are interpreted in different ways depending on what circuits they are fed to. The propagation of such logic levels will be considered later.

CHAPTER 3

EFFECTS OF PHYSICAL FAILURES IN BASIC CMOS CELLS

3.1. Introduction

In this chapter, the effect of physical failures on some basic CMOS cells will be studied. The failure modes in CMOS technology are similar to those in NMOS. It has some additional sources of failures. For example, a "latch-up" occurs under certain conditions between the p-n-p and the n-p-n transistors formed at the junction of the p-well [11]. Latch-up is defined as a high current state accompanied by a low voltage condition. Since the effects of latch-up in CMOS circuits have been extensively studied, they will not be discussed in this thesis. The types of failures that were simulated on CMOS circuits were similar to those for NMOS circuits described in Chapter 2.

It may be noted that since CMOS logic is ratioless, the design rules for layout of such cells are quite different. For NMOS designs, the ratio of the impedance of the pullup transistor to that of the pulldown transistor is usually chosen as approximately 4:1. This is done by choosing the length to width ratio of the depletion transistor about four times greater than that of the enhancement transistor. This is needed to drive the output of the inverter to the required low voltage (about 0.3 V), corresponding to logic "0".

The choice of the dimensions of the transistors in CMOS circuits is motivated by reasons of the current driving capability of the pullup and pulldown transistors rather than the voltage level corresponding to

logic "0". Since the mobility of holes (which are the charge carriers in p-channel transistors) is slightly less than one-half the mobility of electrons (which are the charge carriers in n-channel transistors), the length to width ratio of the p-channel pullup transistor is chosen to be about one-half that of the n-channel pulldown transistor in order to equalize their current driving capabilities and delays.

3.2. CMOS Inverter

The first cell of interest is an inverter. Figure 3.1 shows the different physical failures considered. Here again, the effect of gradual degradation of the failures was studied.

Fault 1: Short between source and gate of n-channel transistor

Referring to Table 3.1, the behavior of the inverter for gradually decreasing resistance values can be seen. For resistance values above 200 K-ohms there is no error. As the resistance decreases to about 40 K-ohms, the voltage level at the output C becomes 3.7 V instead of 5.0 V. This is clearly recognized as a high level by the following inverter. The case where such a faulty inverter drives the gate of a transmission gate will be considered later. The effect of decreasing the resistance is clearly seen. The output becomes higher for logic "0", and lower for logic "1", until R equals 10 K-ohms, below which the output levels are inverted. For values of the resistance below 5 K-ohms, the inverter does not invert the input signal, but the voltage corresponding to the high logic level is around 3.5 V (which corresponds to the logic level "1*").

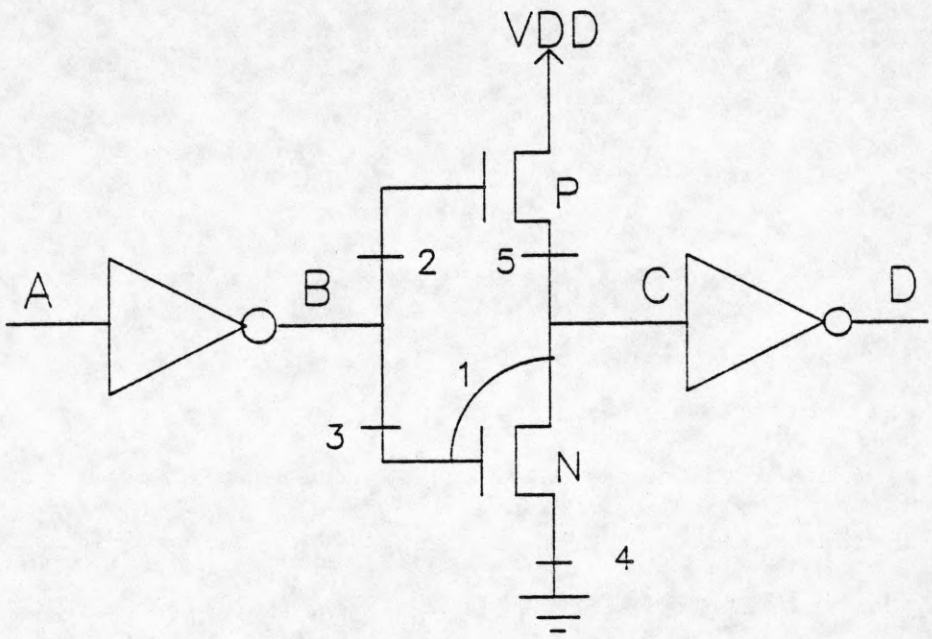


Figure 3.1: CMOS Inverter

Table 3.1: Fault 1 in CMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
2000	0.0	5.0	0.0	5.0	Fault free output.
	5.0	0.0	5.0	0.0	
200	0.0	5.0	0.0	5.0	No error.
	5.0	0.0	5.0	0.0	
40	0.0	4.0	1.0	5.0	Slight error.
	5.0	0.7	3.7	0.0	
15	0.0	3.6	2.2	4.4	Indeterminate.
	5.0	0.94	2.4	1.0	
1	0.0	3.6	3.5	0.0	Error; output inverted.
	5.0	1.0	1.0	5.0	

Fault 2: Floating gate of p-channel device

From Table 3.2, it is noted that the effect of this failure is independent of the value of the resistance. The voltage corresponding to logic "0*" is 1.8 V.

Fault 3: Floating gate of n-channel device

Table 3.3 shows that the effect of this fault is also independent of the value of the resistance. This failure causes sequential behavior. For input A = "0", the output remains at 2.5 V, because both the n-channel and the p-channel devices are on. When A = "1", the output becomes 5.0 V, since only the p-channel device is on. Subsequently as A returns to 0, the output remains at 5.5 V for a long time until the charge leaks away. The p-channel transistor behaves as the pass transistor of a dynamic latch.

Fault 4: Output of several inverters shorted together

It was observed earlier that when the outputs of several NMOS inverters were joined together, the resultant output function was a wired-AND of the individual inverter outputs. However, for CMOS, the situation is quite different. The outputs of four inverters were joined together and the resultant output voltage level was observed for different input combinations. The results are shown in Table 3.4. In Figure 3.2, the output is 5.0 V only when all the inverters outputs are individually 5.0 V. When one of them tries to go to 0.0 V, the overall output goes to 4.0 V. When two of them are low, the resultant output is 1.83 V. When all three inverter outputs are 0.0 V, the overall output is

Table 3.2: Fault 2 in CMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
1000	0.0	5.0	1.8	4.8	No error.
	5.0	0.0	5.0	0.0	
1	0.0	5.0	1.8	4.8	No error.
	5.0	0.0	5.0	0.0	

Table 3.3: Fault 3 in CMOS Inverter

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	COMMENTS
1000	0.0	5.0	2.5	0.7	Sequential behavior.
	5.0	0.0	5.0	0.0	
	0.0	5.0	5.5	0.0	
1	0.0	5.0	2.5	0.7	Sequential behavior.
	5.0	0.0	5.0	0.0	
	0.0	5.0	5.5	0.0	

Table 3.4: Outputs of Four CMOS Inverters Shorted

V(1) (volt)	V(2) (volt)	V(3) (volt)	V(4) (volt)	V(6) (volt)	V(7) (volt)	COMMENTS
0.0	0.0	0.0	0.0	5.0	0.0	Output function
5.0	0.0	0.0	0.0	4.0	0.0	not wired-AND.
5.0	5.0	0.0	0.0	1.8	4.8	
5.0	5.0	5.0	0.0	0.5	4.9	
5.0	5.0	5.0	5.0	0.0	5.0	

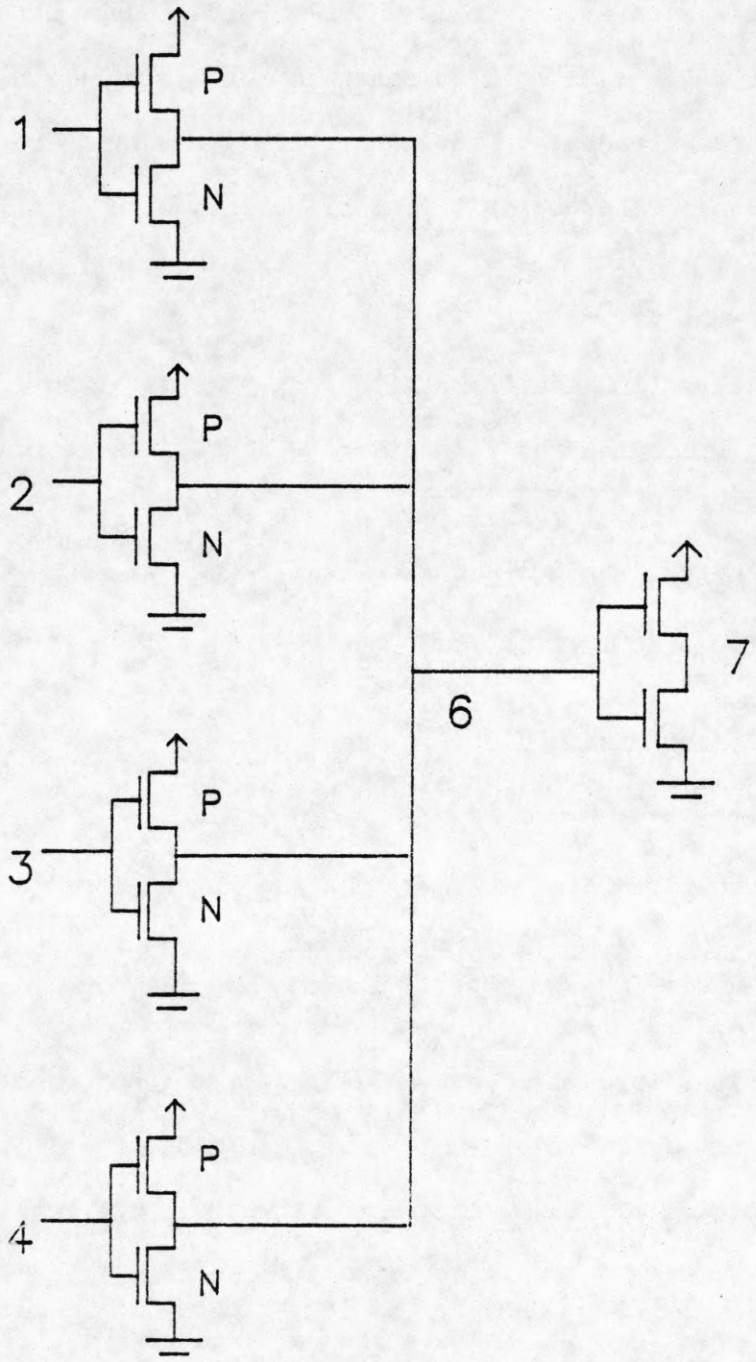


Figure 3.2: CMOS Inverter Outputs Shorted

0.5 V, which is clearly a logic "0". Hence the output function of several CMOS gates shorted together is not a simple wired-AND. Instead, the output is dependent on the number of gates trying to pull the output low, relative to the number trying to pull it high.

3.3. CMOS NAND Gate

The simulation results on a two input CMOS NAND gate, shown in Figure 3.3, will now be discussed. Since the effects of gradual degradation of the physical failures on the NAND gate were very similar to those for the CMOS inverter, only one typical result will be described for each type of fault.

Fault 1: Short between drain and gate of the lower n-channel transistor

This fault was simulated using a resistance of 1 K-ohm between the drain and the gate. Table 3.5 shows that under this fault, the output logic function is altered to a NAND of C and D/ instead of a NAND of C and D. For the input combination, A = "0", B = "0", the output is 3.6 V instead of 0.0 V. For input combination A = "0", B = "1", the output is 1.3 V instead of 5.0 V; there are three voltage levels at the output instead of two. This behavior is similar to that for the NMOS NAND gate for the corresponding fault. The difference is that the intermediate voltage level 3.6 V (which was labeled as "1*" for NMOS) is recognized as logic "1" by any subsequent stage in CMOS. This highly desirable feature is caused by the fact that any voltage level above 3.0 V at the input of a CMOS inverter causes an output less than 0.3 V. This is due

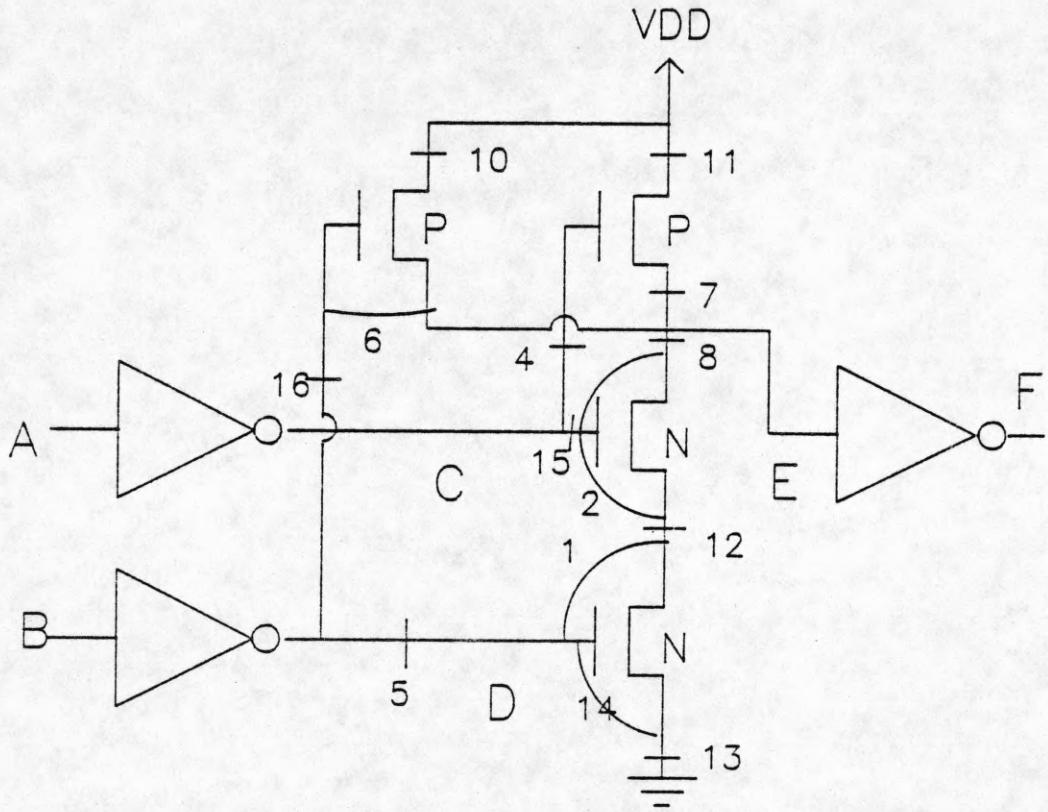


Figure 3.3: CMOS NAND Gate

Table 3.5: Fault 1 in CMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
1000	0.0	0.0	5.0	5.0	0.0	5.0	Fault free.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	0.0	
1	0.0	0.0	5.0	3.6	3.5	0.0	Error.
	5.0	0.0	0.0	3.6	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.6	1.3	5.0	

to the high gain of the transfer characteristics of a CMOS logic gate.

Fault 2: Short between source and gate of the upper n-channel transistor

Table 3.6 shows that this failure results in a fault that can be modeled as the output stuck at 1. This happens because under this type of failure (source to gate short) of any of the n-channel transistors in the NAND gate, the conduction path to ground is cut off, thereby resulting in the output being permanently stuck at 1.

Fault 3: Short between drain and gate of the upper n-channel transistor

Table 3.7 shows the results of the simulation of the failure, using a resistance of 1 K-ohm between the drain and the gate of the faulty device. It is observed that the fault causes the output logic to be altered. There are again three voltage levels at the output. The output function has become equal to C; the gate behaves as if C has been directly connected to the output.

Fault 4: Floating gate of a p-channel transistor

Referring to Figure 3.3, it can be seen that this fault is similar to fault 16. Table 3.8 shows that though there are two voltage levels at the output, both are recognized as logic "1" by any subsequent stage. Hence this fault is equivalent to the output stuck at 1. It is interesting to analyze the case when C = "1", D = "1". Then both the n-channel transistors are on, and under failure, one of the p-channel

Table 3.6: Fault 2 in CMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
1000	0.0	0.0	5.0	5.0	0.0	5.0	Fault free o/p.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	0.0	
1	0.0	0.0	1.9	5.0	5.0	0.0	Output s-a-1.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	5.0	

Table 3.7: Fault 3 in CMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
1000	0.0	0.0	5.0	5.0	0.0	5.0	Fault free o/p.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	0.0	
1	0.0	0.0	4.0	5.0	3.9	0.0	Error.
	5.0	0.0	0.5	5.0	0.6	5.0	
	5.0	5.0	1.3	0.0	1.4	4.9	
	0.0	5.0	5.0	0.0	5.0	0.0	

Table 3.8: Fault 4 in CMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
0	0.0	0.0	5.0	5.0	0.0	5.0	Fault free.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	0.0	
1000	0.0	0.0	5.0	5.0	3.3	0.0	Output s-a-1.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	0.0	

devices is on as well. The output is obtained by a potential division among the "on" resistances of the three transistors, which depends on the relative geometries of the three devices.

Fault 5: Floating gate of the lower n-channel transistor

Figure 3.3 shows that this fault is similar to fault 15. Table 3.9 shows that this fault causes a sequential behavior in the gate. Initially, for the input combination, C = "1", D = "1", the output is 2.0 V, which is recognized as logic "0*". Subsequently when either C or D goes to 0, the output becomes 5.0 V. Now even when both inputs return to 5.0 V, the output remains at 5.0 V because the path to ground is cut off by the faulty transistor. The output will remain "stuck" at 5.0 V for as long as it takes for the charge to leak away from the output through different leakage paths.

Fault 6: Short between source and gate of a p-channel transistor

Table 3.10 shows that this fault, which is similar to fault 3, causes the output to follow D as if it were directly connected. Again, there are three voltage levels at the output.

Fault 8: Drain contact of upper n-channel transistor open

Figure 3.3 shows that this fault is equivalent to open faults in any of the drain or source lines in any n-channel device. This fault results in the output being permanently stuck at 1. Since once the output becomes high, there is no way the output can be pulled low, except through charge leakage while C = "1" and D = "1".

Table 3.9: Fault 5 in CMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
0	0.0	0.0	5.0	5.0	0.0	5.0	Fault free.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	0.0	
	0.0	0.0	5.0	5.0	0.0	5.0	
1000	0.0	0.0	5.0	5.0	2.0	4.6	Error.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	0.0	
	0.0	0.0	5.0	5.0	5.0	0.0	Sequential.

Table 3.10: Fault 6 in CMOS NAND Gate

R (K-ohm)	V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	COMMENTS
1000	0.0	0.0	5.0	5.0	0.0	5.0	Fault free o/p.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	0.0	
	0.0	5.0	5.0	0.0	5.0	0.0	
1	0.0	0.0	5.0	3.8	3.8	0.0	Error.
	5.0	0.0	0.0	5.0	5.0	0.0	
	5.0	5.0	0.0	1.3	1.3	5.0	
	0.0	5.0	5.0	0.6	0.6	5.0	

This fault has been modeled by Wadsack [28,29] as a "stuck-open" fault. It should however be noted that this is only one of the many types of possible faults.

3.4. CMOS NOR Gate

A two input CMOS NOR gate will be considered next. Figure 3.4 shows a two input NOR gate, being driven by inverters and having an inverter as a load. The failure locations are marked on the diagram.

Fault 1: Short between drain and gate of an n-channel transistor

Table 3.11 shows the simulation results for this fault. The failure has been simulated for different resistance values. When the resistance is equal to 15 K-ohms the output remains below 2.1 V for different combinations. Since the threshold of a CMOS gate is around 2.3 V, this voltage level is interpreted as logic "0*". The output of the NOR gate can then be considered to be stuck at "0*".

When the resistance value is decreased to 10 K-ohms, a different logical effect is observed. For the input combination, A = "1", B = "0", the output becomes 2.6 V, which corresponds to the logic level "I". This failure cannot be modeled as a stuck fault at any line.

Fault 2: Short between drain and gate of lower p-channel transistor

From Table 3.12, it may be noted that this failure is very similar to fault 1, because it is equivalent to a short between drain and gate of the other n-channel device. Here again, the failure effects are the same. The resultant output function is a NOR of C/ and D. The failure

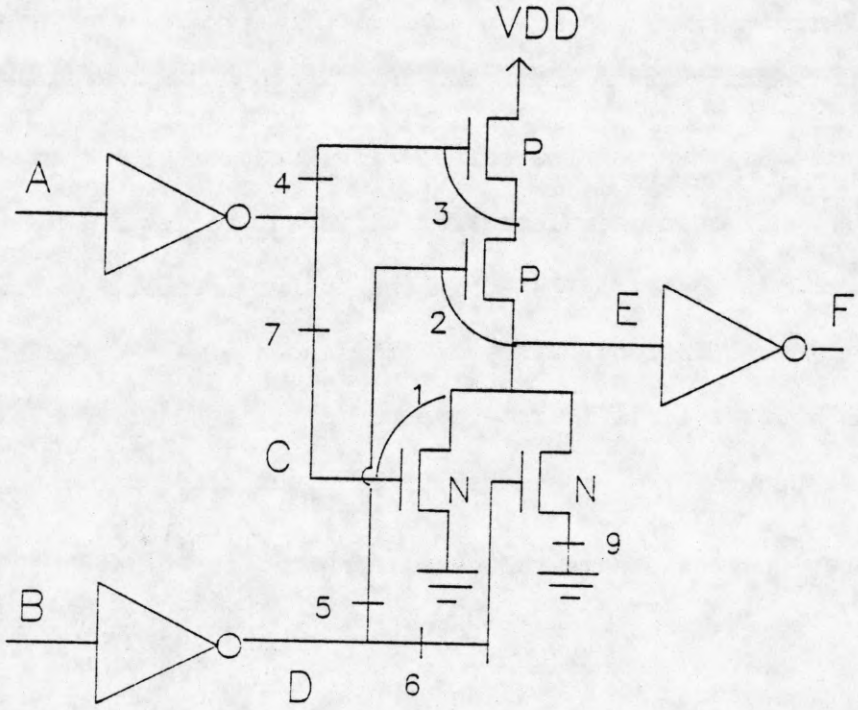


Figure 3.4: CMOS NOR Gate

Table 3.11: Fault 1 in CMOS NOR Gate

R (K-ohm)	V(B) (volt)	V(A) (volt)	V(D) (volt)	V(C) (volt)	V(E) (volt)	COMMENTS
1000	0.0	0.0	5.0	5.0	0.0	Fault free o/p.
	5.0	0.0	0.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	
	0.0	5.0	5.0	0.0	0.0	
15	0.0	0.0	5.0	2.9	0.9	Output s-a-0.
	5.0	0.0	0.0	3.6	2.1	
	5.0	5.0	0.0	0.7	1.8	
	0.0	5.0	5.0	0.0	0.0	
10	0.0	0.0	5.0	2.6	1.1	Function changed.
	5.0	0.0	0.0	3.6	2.6	
	5.0	5.0	0.0	0.8	1.5	
	0.0	5.0	5.0	0.0	0.0	
0	0.0	0.0	5.0	1.7	1.8	Same as above.
	5.0	0.0	0.0	3.6	3.6	
	5.0	5.0	0.0	0.7	0.7	
	0.0	5.0	5.0	0.0	0.0	

Table 3.12: Fault 2 in CMOS NOR Gate

R (K-ohm)	V(B) (volt)	V(A) (volt)	V(D) (volt)	V(C) (volt)	V(E) (volt)	COMMENTS
1000	0.0	0.0	5.0	5.0	0.0	Fault free o/p.
	5.0	0.0	0.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	
	0.0	5.0	5.0	0.0	0.0	
1	0.0	0.0	1.7	5.0	1.7	Output function changed
	5.0	0.0	0.0	5.0	0.0	
	5.0	5.0	0.6	0.0	0.6	
	0.0	5.0	3.6	0.0	3.6	

cannot be modeled as a stuck fault in any line.

Fault 3: Short between gate and source of upper p-channel transistor

Table 3.13 shows that this failure results in the output remaining below 1.4 V. This fault can therefore be modeled as the output line stuck at "0".

Fault 4: Floating gate of upper p-channel transistor

This failure was simulated by opening the connection to the gate and placing a small resistance between the gate and the ground. This had to be done because SPICE does not allow floating nodes, owing to convergence problems. Table 3.14 shows that there is no error in the output logic function, but there is a timing error. From the SPICE results in Figure 3.5 it can be seen that it takes approximately 6 times the time to charge up to logic "1" compared to the fault free gate. Such timing errors may be of importance in any system expecting a signal to be present within a certain time.

Fault 5: Floating gate of lower p-channel transistor

Table 3.15 shows that this failure is very similar to fault 4. There is no logical error, but there is a timing error; it takes about 6 times as long to charge up to logic "1" from logic "0".

Fault 6: Floating gate of an n-channel transistor

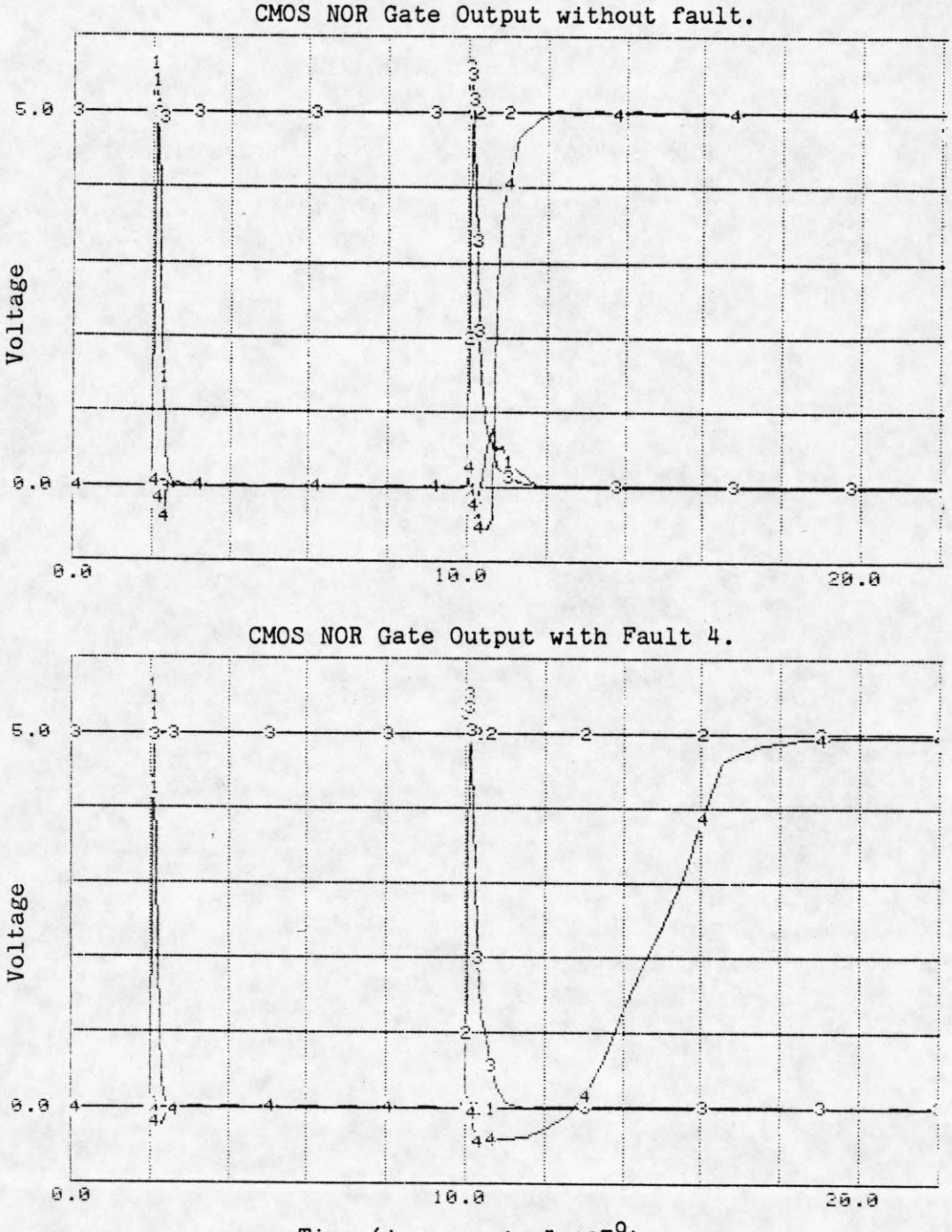
This results in an extremely interesting fault; the circuit becomes sequential in nature. Table 3.16 shows the results of the simulation. For the input combination, A = "1", B = "0", the output retains its

Table 3.13: Fault 3 in CMOS NOR Gate

R (K-ohm)	V(B) (volt)	V(A) (volt)	V(D) (volt)	V(C) (volt)	V(E) (volt)	COMMENTS
1000	0.0	0.0	5.0	5.0	0.0	Fault free o/p.
	5.0	0.0	0.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	
	0.0	5.0	5.0	0.0	0.0	
1	0.0	0.0	5.0	5.0	0.0	Output s-a-0.
	5.0	0.0	0.0	3.8	1.4	
	5.0	5.0	0.0	1.0	1.0	
	0.0	5.0	5.0	0.0	0.0	

Table 3.14: Fault 4 in CMOS NOR Gate

R (K-ohm)	V(B) (volt)	V(A) (volt)	V(D) (volt)	V(C) (volt)	V(E) (volt)	COMMENTS
1000K	0.0	0.0	5.0	5.0	0.0	No logical error
	5.0	0.0	0.0	5.0	0.0	but a timing error
	5.0	5.0	0.0	0.0	5.0	takes 6 times more
	0.0	5.0	5.0	0.0	1.0	charge-up time.



LEGEND: Node C in Figure 3.4 ..... 3 in plot
Node D in Figure 3.4 ..... 1 in plot
Node E in Figure 3.4 ..... 4 in plot

Figure 3.5: SPICE Simulation Results Showing Timing Error for Fault 4

Table 3.15: Fault 5 in CMOS NOR Gate

R (K-ohm)	V(B) (volt)	V(A) (volt)	V(D) (volt)	V(C) (volt)	V(E) (volt)	COMMENTS
1000K	0.0	0.0	5.0	5.0	0.0	Similar timing error.
	5.0	0.0	0.0	5.0	0.0	Takes 6 times more
	5.0	5.0	0.0	0.0	5.0	charge-up time.
	0.0	5.0	5.0	0.0	0.0	

Table 3.16: Fault 6 in CMOS NOR Gate

R (K-ohm)	V(B) (volt)	V(A) (volt)	V(D) (volt)	V(C) (volt)	V(E) (volt)	COMMENTS
1	0.0	5.0	5.0	0.0	0.0	Circuit becomes
	0.0	0.0	5.0	5.0	0.0	sequential.
	5.0	0.0	0.0	5.0	0.0	
	5.0	5.0	0.0	0.0	5.0	
	0.0	5.0	5.0	0.0	5.4	Logic "1" retained.

previous value. The table shows the retention of logic "1" as 5.4 V (the voltage is greater than 5.0 V because of the capacitance of the output node which is charged up to that voltage by the transient). This behavior can be explained as follows. Since the gate of one n-channel device is floating, that device will be constantly off. Consider the case when D is low. When C is high the output is low. Now when C becomes low the output becomes high, driving any capacitive load to 5.0 V. When C subsequently returns to logic "0", the output remains at logic "1". This causes the sequential behavior. This is another example of a "stuck-open" fault.

3.5. CMOS Transmission Gate

A CMOS transmission gate consists of a p-channel and an n-channel transistor, connected back to back, such that their drains and sources are together. The signal, controlling the opening and closing of the transmission gate, is applied to the gate of the n-channel device while its complement is applied to the gate of the gate of the p-channel device. This is done so that depending on the input to the transmission gate, one of the devices operates in the drain-loaded mode, thereby driving the output to 5.0 V or 0.0 V. This is where the CMOS transmission gate is different from the corresponding NMOS gate; the latter can pull the output down to 0.3 V, but cannot drive the output to a voltage level higher than VDD-Vto, where Vto is the threshold voltage of the pass transistor.

Figure 3.6 shows the circuit considered for simulation. Table 3.17 shows the results for a fault free circuit. All inputs are driven by

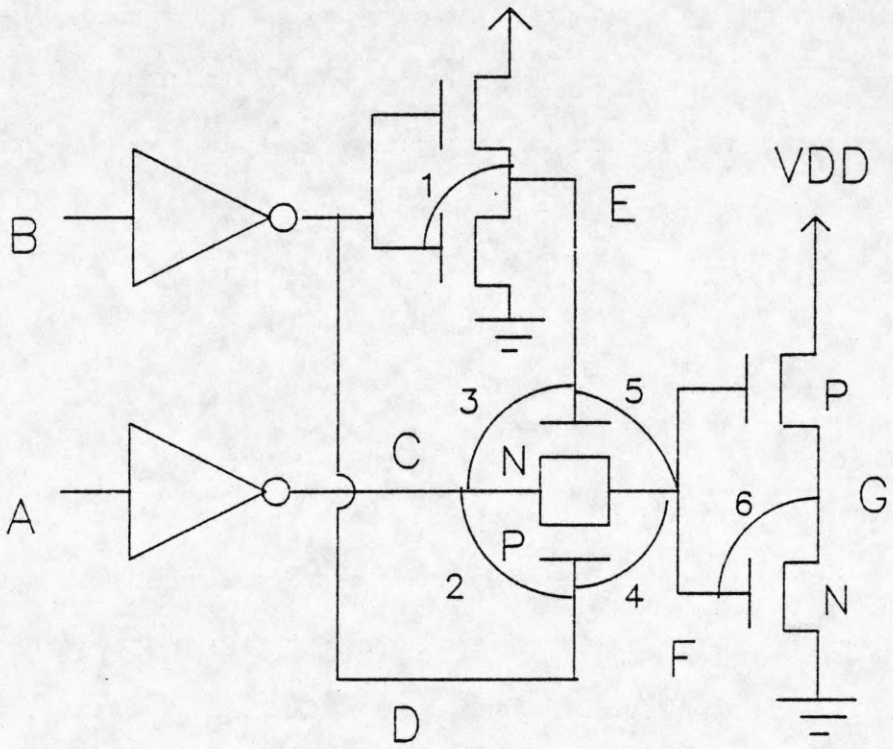


Figure 3.6: CMOS Transmission Gate

Table 3.17: Fault-free Behavior of CMOS Transmission Gate

V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	V(G) (volt)	COMMENTS
0.0	0.0	5.0	5.0	0.0	0.0	5.0	Previous value
5.0	0.0	0.0	5.0	0.0	0.0	5.0	of F retained.
5.0	5.0	0.0	0.0	5.0	0.0	5.0	New values
0.0	5.0	5.0	0.0	5.0	5.0	0.0	stored.
0.0	0.0	5.0	5.0	0.0	5.0	0.0	Previous value
5.0	0.0	0.0	5.0	0.0	5.0	0.0	of F retained.

inverters.

Fault 1: Short between drain and gate of n-channel transistor

Table 3.18 shows that output of the transmission gate attains voltage levels other than 0.0 V and 5.0 V. Due to the failure, the gates of both the n- and p-channel devices receive the same signal levels. With the control signal at 1.0 V, the n-channel device does not conduct at all, but the p-channel device conducts well if the input C equals 5.0 V, driving the output to 5.0 V. However, if the input is 0.0 V, the output is pulled to 2.0 V instead of 0.0 V because the threshold voltage drop of the p-channel device is -1.0 V.

With the control signal at 3.6 V, the p-channel device remains off, but the n-channel device is on. If C equals 0.0 V, then the output is pulled down to 0.0 V irrespective of the value of the previous value of the output stored in the capacitance of the gate of the output inverter. When the input is 5.0 V, the output depends on its previous value. If the previous value is 0.0 V, then the output slowly rises to 2.5 V; however, if the previous value is 5.0 V, it remains at that value.

Fault 2: Short between source and gate of p-channel pass transistor

Table 3.19 shows that this failure results in the output of the transmission gate remaining within 2.1 V, which is interpreted as logic "0*". Hence this fault can be modeled as the output being stuck at "0*". However, the input to the gate of the n-channel device is affected as well. It is assigned the "wired-AND" of the outputs of the two buffers driving the gate and source of the n-channel device.

Table 3.18: Fault 1 in CMOS Transmission Gate

V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	V(G) (volt)	COMMENTS
0.0	0.0	5.0	3.6	3.6	2.5	0.7	Indeterminate.
5.0	0.0	0.0	3.6	3.6	0.0	5.0	
5.0	5.0	0.0	1.0	1.0	2.0	4.7	
0.0	5.0	5.0	1.0	1.0	5.0	0.0	
0.0	0.0	5.0	3.6	3.6	5.0	0.0	
5.0	0.0	0.0	3.6	3.6	0.0	5.0	Memory loss.

Table 3.19: Fault 2 in CMOS Transmission Gate

V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	V(G) (volt)	COMMENTS
0.0	0.0	5.0	5.0	0.0	2.1	4.5	Output s-a-0.
5.0	0.0	1.8	1.8	4.8	1.8	4.8	
5.0	5.0	0.0	0.0	5.0	0.0	5.0	
0.0	5.0	1.8	1.8	4.8	1.8	4.8	
0.0	0.0	5.0	5.0	0.0	2.1	4.5	
5.0	0.0	1.8	1.8	4.8	1.8	4.8	

Fault 3: Short between source and gate of n-channel pass transistor

From Table 3.20, it can be seen that the output has two voltage levels, 2.0 V and 5.0 V. The source C of the transmission gate is shorted to the gate D of the n-channel device. When C = "1", and D = "1", the output becomes 5.0 V. However, when C is changed to "0", the resultant voltage level becomes 1.85 V, and the output is pulled to 2.0 V. When C = "0", and D = "0", the resultant voltage level at C and D is 0.0 V; hence, the previous value of the output is retained. When C = "1", and D = "0", the resultant voltage level becomes 1.85 V; the previous value of the output is retained.

Fault 4: Short between drain and gate of p-channel pass transistor

Table 3.21 shows that this failure results in the direct connection of D to F. Then G follows B after two inverter delays, similar to the corresponding fault in the NMOS transmission gate.

Fault 5: Short between drain and gate of n-channel pass transistor

This fault is similar to fault 4 above, except that E is directly connected to F. Table 3.22 shows that G follows D after two inverter delays.

Fault 6: Short between drain and gate of n-channel transistor

Table 3.23 shows that when the transmission gate is switched off, the output is stuck at 2.4 V, which corresponds to the indeterminate logic level "I" since the threshold of a CMOS logic gate is around 2.4 V. When the transmission gate is on, and the input C is 0.0 V, the

Table 3.20: Fault 3 in CMOS Transmission Gate

V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	V(G) (volt)	COMMENTS
0.0	0.0	1.8	5.0	1.8	2.1	4.5	Output function
5.0	0.0	0.0	5.0	0.0	2.1	4.5	altered.
5.0	5.0	1.8	0.0	1.8	2.0	4.6	
0.0	5.0	5.0	0.0	5.0	5.0	0.0	
0.0	0.0	1.8	5.0	1.8	5.0	0.0	
5.0	0.0	0.0	5.0	0.0	5.0	0.0	

Table 3.21: Fault 4 in CMOS Transmission Gate

V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	V(G) (volt)	COMMENTS
0.0	0.0	5.0	5.0	0.0	5.0	0.0	V(6) follows
5.0	0.0	0.0	5.0	0.0	5.0	0.0	V(4).
5.0	5.0	0.0	0.0	5.0	0.0	5.0	
0.0	5.0	3.3	1.2	5.0	1.2	5.0	
0.0	0.0	5.0	5.0	0.0	5.0	0.0	
5.0	0.0	0.0	5.0	0.0	5.0	0.0	

Table 3.22: Fault 5 in CMOS Transmission Gate

V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	V(G) (volt)	COMMENTS
0.0	0.0	5.0	5.0	0.0	0.0	5.0	V(6) follows
5.0	0.0	0.0	5.0	0.0	5.0	0.0	V(5).
5.0	5.0	0.9	0.0	3.7	3.7	0.0	
0.0	5.0	5.0	0.0	5.0	5.0	0.0	
0.0	0.0	5.0	5.0	0.0	0.0	5.0	
5.0	0.0	0.0	5.0	0.0	0.0	5.0	

Table 3.23: Fault 6 in CMOS Transmission Gate

V(A) (volt)	V(B) (volt)	V(C) (volt)	V(D) (volt)	V(E) (volt)	V(F) (volt)	V(G) (volt)	COMMENTS
0.0	0.0	5.0	5.0	0.0	2.4	2.4	Indeterminate.
5.0	0.0	0.0	5.0	0.0	2.4	2.4	
5.0	5.0	0.6	0.0	5.0	1.4	1.4	
0.0	5.0	4.4	0.0	5.0	3.0	3.0	
0.0	0.0	5.0	5.0	0.0	2.4	2.4	
5.0	0.0	0.0	5.0	0.0	2.4	2.4	

output becomes 1.4 V. But, when C equals 5.0 V, the output is pulled up to 3.0 V.

3.6. Conclusion

In this chapter, we have discussed some of the effects of physical failures in CMOS circuits. It must be noted that the failures simulated were identical to those in NMOS technology. In addition, CMOS has certain special fault conditions such as the "latch-up", which results in a high conductance path between VDD and ground. Such failures are said to be catastrophic in nature; meaning that these faults will result in conditions that cause more failures such as opens in the power or ground lines to occur. They are therefore easy to detect.

CHAPTER 4

FUNCTIONAL FAULT MODELS

4.1. Introduction

In this chapter, the effects of physical failures on some functional modules, typically used in NMOS designs, will be presented. In view of the increasing complexity of VLSI circuits today, gate level testing of complex modules is becoming more and more difficult. Efforts are gradually being directed towards obtaining functional level fault models, which help to reduce the complexity of test generation. In this approach, functional units such as decoders, adders, and programmable logic arrays are considered as primitive elements. Instead of considering faults at the device level (such as opens or shorts in lines), or at the gate level (such as inputs and outputs of logic gates stuck at "0" or "1"), the effects of the faults are viewed at a functional level.

In general, without having any idea about the kinds of physical failures one has to be prepared for, it is often impossible to know what functional changes can occur under failure. Testing such an unit involves exhaustive testing of the functional block for all sets of inputs for combinational units, and all sequences of inputs for sequential units. Such exhaustive testing of functional blocks is not always feasible due to the limited controllability and observability of a VLSI module, and also due to the immense number of input patterns required to test such units.

Our motivation behind the study of faults in MOS circuits by simulating the physical failures at the circuit level is to obtain results which will be useful in the functional fault modeling of various modules. Studies have been performed on modules that have a certain amount of structural regularity. The results on programmable logic arrays and decoders will now be discussed.

4.2. Programmable Logic Array

Programmable logic arrays (PLA) are being increasingly used in VLSI circuits today to realize arbitrary logic functions in a very short design time due to their tremendous structural regularity. A lot of effort has therefore been directed towards developing fault models for PLAs. However, most approaches have been based on fault models which do not cover a lot of physical failures. From our results of simulation on typical PLAs used in NMOS designs, an existing fault model was validated. However, the fault model is subject to a design restriction.

Figure 4.1 shows the circuit diagram of a PLA, typically used in NMOS designs using a two-phase clocking scheme. The diagram includes the input and output registers. It is assumed that only the inputs are externally available. Their complements have to be derived by passing them through inverters. The inputs and their complements, stored during the phase 1 clock in input registers, are run vertically through a matrix of circuit elements called the AND plane. The AND plane generates specific logic combinations (product terms) of the inputs and their complements. The outputs of the AND plane leave at right angles to its inputs and run horizontally through another matrix called the OR plane.

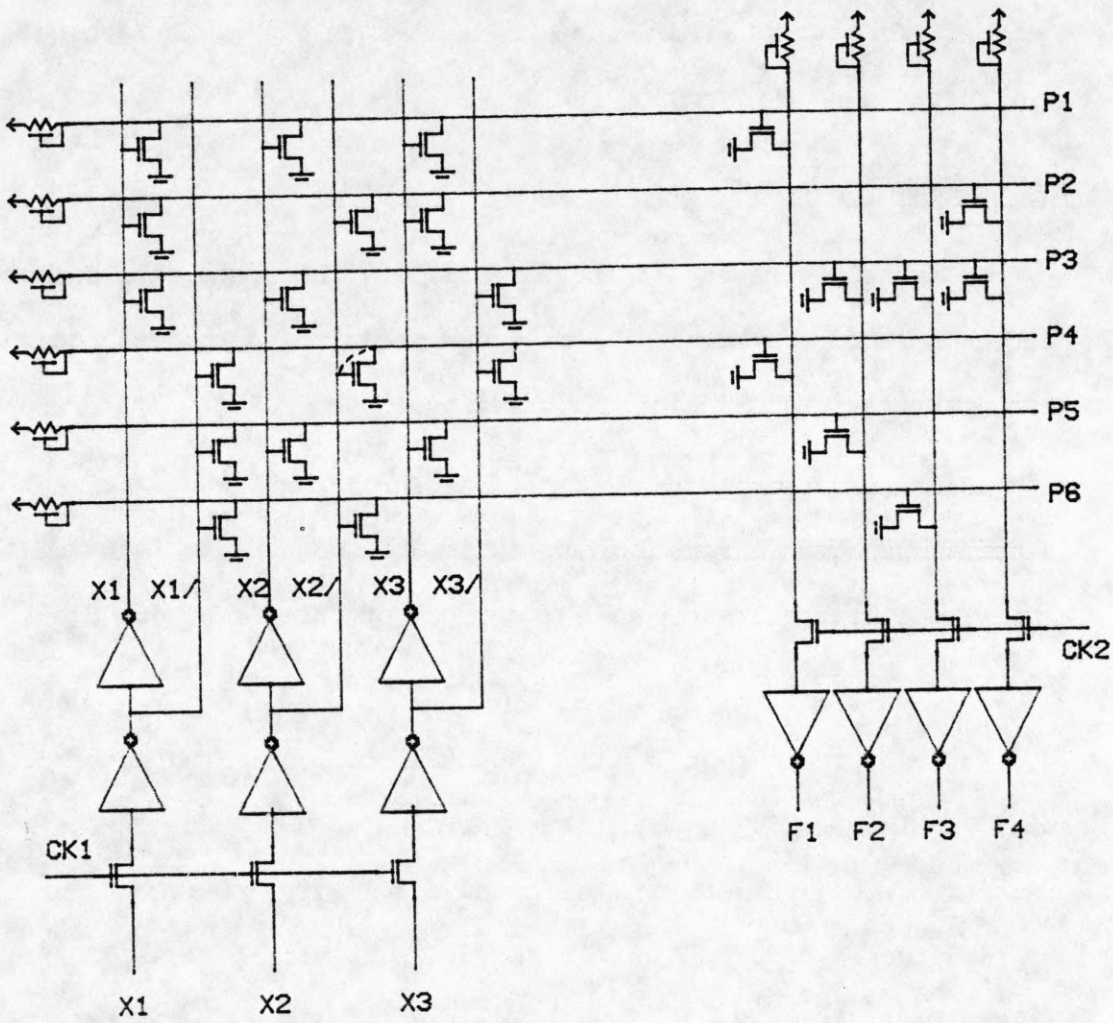


Figure 4.1: NMOS NOR-NOR PLA

The outputs of the OR plane are the desired functions, which are fed vertically and are stored in the output registers during the phase 2 clock.

The two level matrix arrays are formed by NOR gates or NAND gates. However, for large PLAs, the NOR-NOR structure is generally more preferred.

Fault Model for PLAs

An exhaustive study of the effects of the physical failures on different PLAs revealed that, under certain design restrictions, all single faults in PLAs result in unidirectional errors.

An unidirectional error is defined for a sequence of binary digits (bits). Due to errors anywhere in the sequence of 0's and 1's, if either some of the 0's become 1's or some of the 1's become 0's (but not both), then the error is said to be unidirectional.

However, without any design constraint, this property does not hold. In Figure 4.1, the indicated failure (a short between the gate and drain of the enhancement transistor of a NOR gate) results in a non-unidirectional error. For the input combination $X_1X_2X_3$ = 100, the outputs $F_1F_2F_3F_4$ become 0010 instead of 0100 due to a rather strange feedback. Since $X_3/$ = 1, P_4 = 0. This forces the $X_2/$ input of the faulty NOR gate to be pulled down to a 0. Since $X_2/$ is passed through an inverter to derive X_2, the latter is forced to a 1. Hence P_6 goes from 0 to a 1, whereas P_5 goes from 1 to a 0. This forces F_2 to go from 1 to 0 and F_3 from 0 to 1, giving rise to a non-unidirectional error.

Design Restrictions on PLAs

The design restrictions which will force physical failures to cause only unidirectional errors are quite simple, and will not lead to much overhead. They are shown in Figure 4.2 and described below:

(1) Buffers have be placed between the primary inputs and the AND plane inputs of the PLA to avoid any feedback.

(2) There must be a way to detect a stuck fault at the primary input line before the fanout point.

Under those restrictions, all single faults in a PLA will result in unidirectional errors.

4.3. Decoder

One of the most widely used units in VLSI circuits is a decoder, which has a considerable amount of structural regularity. Two common ways of realizing decoders in NMOS technology are the NAND form and the NOR form, both of which were studied.

The following representation is used to describe the behavior of a faulty decoder. For a 2-input decoder, since there are two address lines, the rows in the table refer to the 2-bit binary address of the line which would be selected by a fault-free decoder. The columns in the table refer to the 2-bit binary address of the line that is actually selected by the faulty decoder. For the 2-bit input combination, corresponding to a particular row, an entry in a column is marked "1" if a line, that corresponds to the column address, is selected; otherwise, it is marked "0".

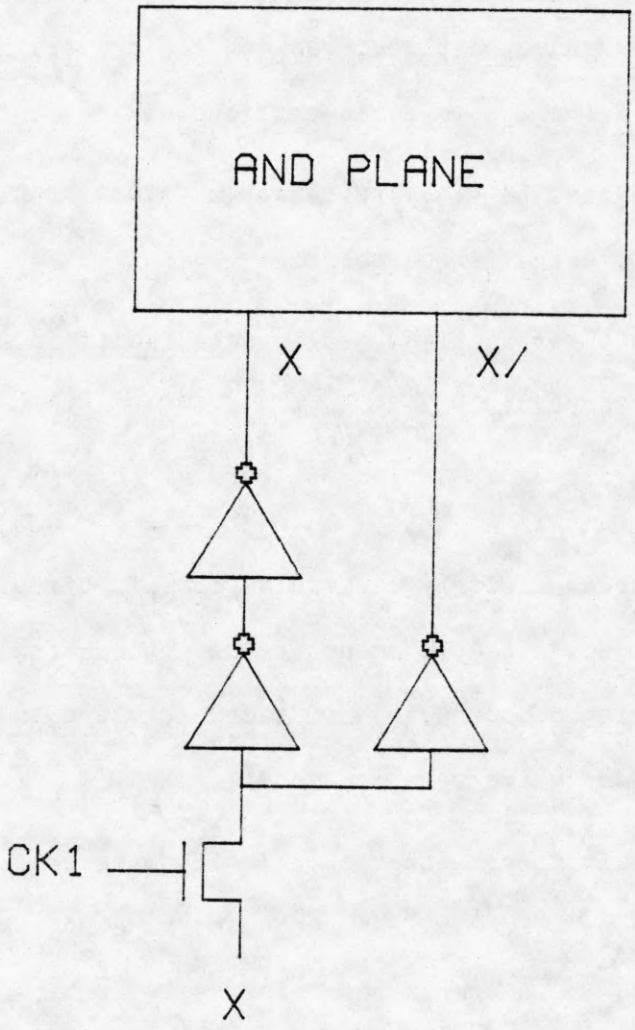


Figure 4.2: Design Restriction on PLAs

Some typical results on a NAND 2-to-4 decoder, shown in Figure 4.3, are listed in Table 4.1. All single physical failures under our assumed failure modes were simulated.

For a NAND decoder, the "selection" of a line means that that line has been pulled down to logic zero. The entry "I" refers to an indeterminate logic level at the output of the decoder. Depending on what stage is being driven, that will be recognized as a "0" or a "1".

Figure 4.4 shows a 2-to-4 decoder NMOS circuit made up of NOR gates. The corresponding fault locations can be referred from Figure 4.4. The representation of the behavior of the NOR decoder, shown in Table 4.2, is similar to that for the NAND decoder, except that here, the "selection" of a line means that the corresponding line becomes physically high. For the 2-bit input combination corresponding to a row address we mark the entry under a column to be "1", if the line corresponding to the column address is selected, otherwise it is marked "0".

Fault Model for Decoders

The following fault model was developed for any n-input NMOS decoder having the same structure as shown in the Figures 4.3 and 4.4. All single faults can be covered by the following classes of functional faults:

(1) $f(L_i/L_j)$: Instead of line L_i, line L_j is selected.

(2) $f(L_i/L_i+L_j)$: In addition to L_i, L_j is selected.

(3) $f(L_i/0)$: None of the lines are selected.

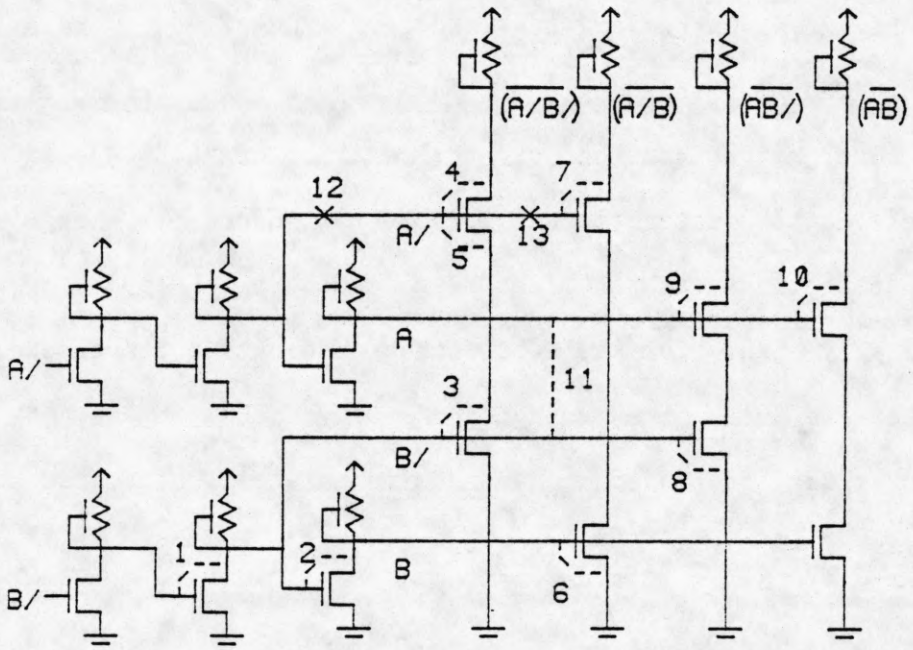


Figure 4.3: NMOS NAND Decoder

Table 4.1: Effect of Failures on a NAND Decoder without any Design Restrictions

Fault 1:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	0	0	0	1	$f(L_0/L_1)$
10	0	0	1	0	$f(L_2/L_3)$
11	0	1	0	0	$f(L_3/L_2)$
01	1	0	0	0	$f(L_1/L_0)$

Fault 2:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	1	0	0	1	$f(L_0/L_0+L_1)$
10	0	1	1	0	$f(L_2/L_2+L_3)$
11	0	0	0	0	$f(L_3/0)$
01	0	0	0	0	$f(L_1/0)$

Fault 3:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	I	0	0	1	$f(L_0/L_0+L_1)$
10	0	I	1	0	$f(L_2/L_2+L_3)$
11	0	0	1	0	$f(L_3/L_3)$
01	1	0	0	1	$f(L_1/L_1+L_0)$

Fault 4:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	I	0	0	0	$f(L_0/L_0)$
10	1	1	0	0	$f(L_2/L_2+L_0)$
11	1	0	1	0	$f(L_3/L_3+L_0)$
01	0	0	0	1	$f(L_1/L_1)$

Table 4.1 (contd.)

Fault 7:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	1	0	0	0	$f(L_0/L_0)$
10	0	1	0	1	$f(L_2/L_2+L_1)$
11	0	0	1	1	$f(L_3/L_3+L_1)$
01	0	0	0	I	$f(L_1/L_1)$

Fault 9:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	1	1	0	0	$f(L_0/L_0+L_2)$
10	0	I	0	0	$f(L_2/L_2)$
11	0	0	1	0	$f(L_3/L_3)$
01	0	1	0	1	$f(L_1/L_1+L_2)$

Fault 11:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	0	0	0	1	$f(L_0/L_1)$
10	0	1	0	0	$f(L_2/L_2)$
11	0	0	0	0	$f(L_3/0)$
01	1	0	0	1	$f(L_1/L_1+L_0)$

Fault 13:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	1	0	0	0	$f(L_0/L_0)$
10	0	1	0	0	$f(L_2/L_2)$
11	0	0	1	0	$f(L_3/L_3)$
01	0	0	0	0	$f(L_1/0)$

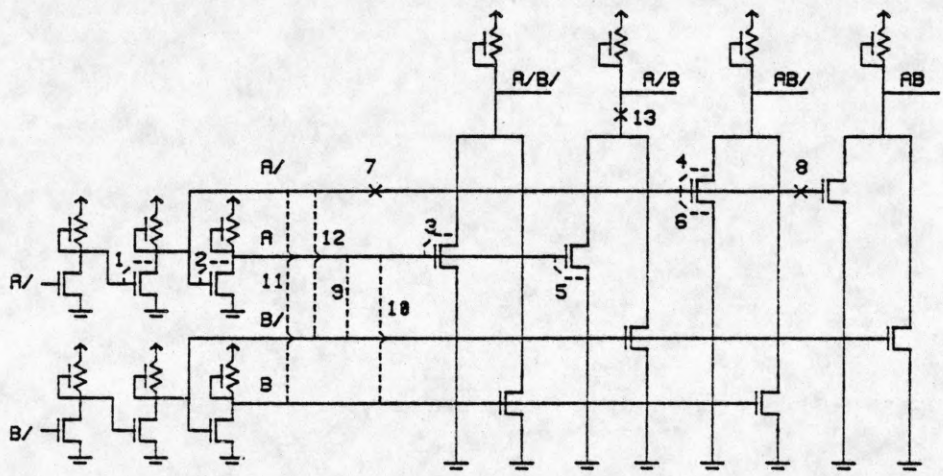


Figure 4.4: NMOS NOR Decoder

Table 4.2: Effects of Failures on a NOR Decoder Without Any Design Restrictions

Fault 1:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	0	1	0	0	$f(L_0/L_2)$
10	1	0	0	0	$f(L_2/L_0)$
11	0	0	0	1	$f(L_3/L_1)$
01	0	0	1	0	$f(L_1/L_3)$

Fault 2:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	0	0	0	0	$f(L_0/0)$
10	1	1	0	0	$f(L_{2/L_2+L_0})$
11	0	0	1	1	$f(L_3/L_3+L_1)$
01	0	0	0	0	$f(L_1/0)$

Fault 3:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	0	0	0	0	$f(L_0/0)$
10	1	0	0	0	$f(L_2/L_0)$
11	0	0	1	1	$f(L_3/L_3+L_1)$
01	0	0	0	1	$f(L_1/L_1)$

Fault 4:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	1	1	0	0	$f(L_0/L_0+L_2)$
10	0	0	0	0	$f(L_2/0)$
11	0	0	1	0	$f(L_3/L_3)$
01	0	0	1	0	$f(L_1/L_3)$

Table 4.2 (contd.)

Fault 7:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	1	1	0	0	$f(L_0/L_0+L_2)$
10	0	1	0	0	$f(L_2/L_2)$
11	0	0	1	0	$f(L_3/L_3)$
01	0	0	1	1	$f(L_1/L_1+L_3)$

Fault 8:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	1	0	0	0	$f(L_0/L_0)$
10	0	1	0	0	$f(L_2/L_2)$
11	0	0	1	0	$f(L_3/L_3)$
01	0	0	1	1	$f(L_1/L_1+L_3)$

Fault 9:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	0	0	0	1	$f(L_0/L_1)$
10	0	1	0	0	$f(L_2/L_2)$
11	0	0	1	1	$f(L_3/L_3+L_1)$
01	0	0	0	1	$f(L_1/L_1)$

Fault 10:

Address of desired selection	Address of actual selection 00	10	11	01	COMMENTS
00	1	0	0	0	$f(L_0/L_1)$
10	1	1	0	0	$f(L_2/L_2+L_0)$
11	0	0	1	0	$f(L_3/L_3)$
01	1	0	0	1	$f(L_1/L_1+L_0)$

However, if the inputs to the NOR or NAND gates of the decoder are buffered as described for the PLA above (Figure 4.2), then the fault classes collapse to:

(1) $f(L_i/L_i+L_j)$.

(2) $f(L_i/0)$.

The above fault model is true for both NAND and NOR decoders. However, NOR decoders satisfy another condition under certain design restrictions.

For NOR decoders, further refinement is obtained by relating the L_j's that can be selected in addition to L_i.

Definition 1:

Zero Count of a line selected by a decoder refers to the number of zeros in its binary address.

Definition 2:

An error in a decoder is said to be Zero Count Detectable (ZCD) if the Zero Count of the required selection differs from the Zero Count of a faulty selection.

Example:

For the 2-to-4 decoder, if instead of line L_1 (whose zero count is 1), line L_0 (whose zero count is 2) is selected, the error would be Zero Count Detectable. However, if instead of line L_1 (whose zero count is 1), line L_2 (whose zero count is also 1) is selected, the error would not be Zero Count Detectable.

It may be noted that an output line stuck at 1 can cause the property of Zero Count Detectability not to hold for a NOR decoder. For example, if the line L_1, corresponding to A/B (whose zero count is 1) is somehow stuck at 1, then when the line L_2, corresponding to AB/, should be selected, the former is selected as well. This would give rise to an error that is not ZCD.

Such failures must not be allowed to occur. A physical failure that can give rise to such a fault can be considered, along with a corresponding design rule that avoids such a failure. From Figure 4.4, it may be noted that if there is an open in the line between the pullup transistor source and the drain of a pulldown enhancement device (Fault 13), then the output can remain stuck at 1. However, from the typically observed NMOS failure modes, listed in Chapter 2, it is seen that opens in diffusion lines are unlikely. Hence, to prevent the possibility of such a failure, it might be necessary to enforce a design restriction that the line connecting those two critical points should be made of diffusion lines; also, those lines should be wide enough to prevent the occurrence of such an open failure.

Our results have shown that all other single faults (i.e., faults due to single physical failures mentioned earlier) in NMOS NOR decoders result in ZCD errors subject to the design restriction mentioned.

The property of Zero Count Detectability does not hold for NAND decoders in NMOS.

For example, for fault 7, instead of the line whose address is "10"(with zero count = 1), both the lines with addresses "10" and "01"

(whose zero count equals 1) are selected. Also for fault 9, instead of the line "01"(zero count = 1), both lines "01" and "10" (zero count = 1) are selected. Both these examples show the cases of non-zero-count-detectable faults in NAND decoders.

Design Restrictions on NMOS Decoders

(1) The inputs to the NAND or NOR gates should be buffered, as shown in Figure 4.5.

(2) There has to be a way of detecting a stuck-fault at any primary input of the decoder.

(3) For NOR decoders, an additional restriction mentioned above (preventing stuck-at-1 faults at the output of a decoder), provides Zero Count Detectability. There is no corresponding design rule for NAND decoders to make them Zero Count Detectable.

The property of Zero Count Detectability can be used in designing concurrent error detection into decoders by using codes such as the Berger Code.

4.4. Conclusion

The above examples demonstrate the usefulness of our approach to obtain functional fault models by simulating physical failures at the circuit level. Conventional approaches towards developing such models have relied on assumptions that have had relatively little correspondence with physical failures. A study of the effects of physical failures at the circuit level can provide accurate fault models

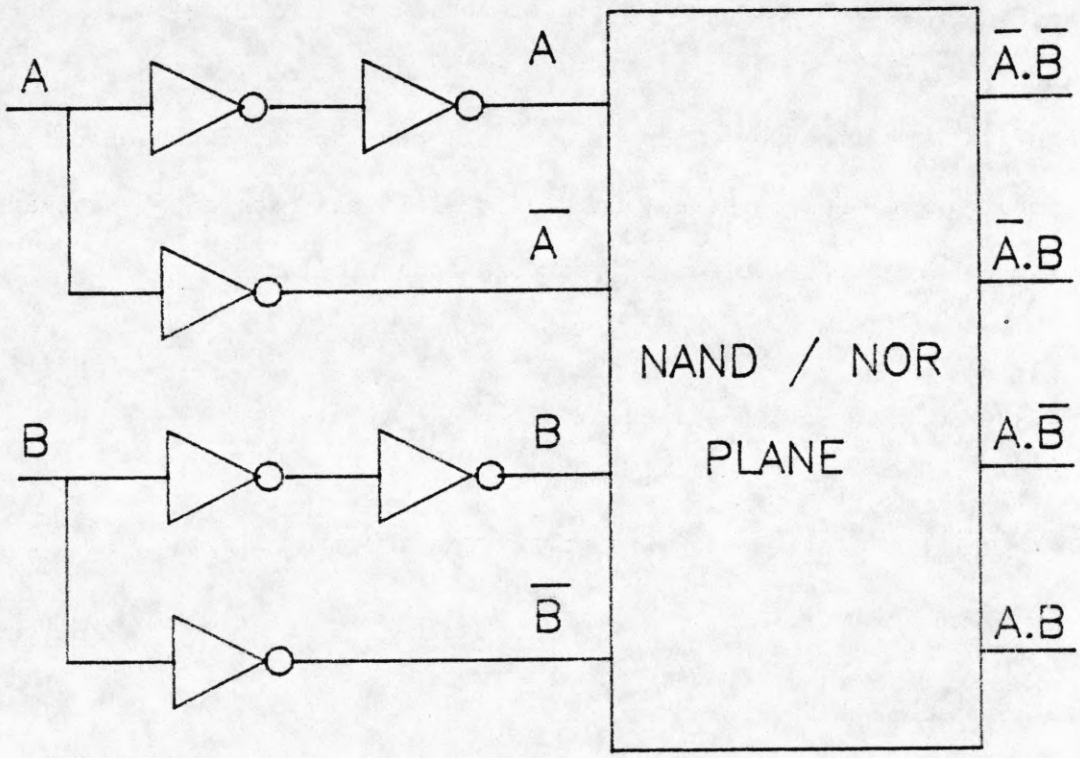


Figure 4.5: Design Restriction on NMOS Decoders

and also suggest certain designs for testability. However, such an approach has two limitations. Firstly, it is implementation dependent. However, since this approach is geared towards designs for testability, such studies may be important to logic designers. Secondly, detailed studies of failures at the circuit level can be very time consuming for complex VLSI modules.

A two-step approach was therefore used. Firstly, detailed studies were performed at the circuit level to investigate the effects of physical failures on small modules. Secondly, on the basis of these results an abstract logical model of the effects of failures was constructed which enabled quick and relatively accurate estimation of the behavior of complex VLSI modules under failures.

The logical model was developed in the form of a multi-valued algebra which will be discussed in the next chapter.

CHAPTER 5

A MULTI-VALUED ALGEBRA

5.1. Introduction

An accurate understanding of physical failures on digital circuits may be obtained by studying their effects at the circuit level. Such studies on MOS circuits have revealed the existence of abnormal failure modes, as shown in Chapters 2, 3, and 4. Unfortunately, such detailed studies are not practical for complex VLSI modules.

It is therefore desirable to have logical models for describing the behavior of MOS circuits under failures. Classical switching theory fails to account for some key structural and logical properties of MOS circuits, such as the bidirectional transmission gate, and tri-state busses. Switch level simulation models [4,14,17] have emerged as efficient logic design tools for MOS circuits. However, these models cannot be used to describe the behavior of MOS circuits under physical failures typically observed in the field [31] (which are summarized in Table 5.1).

In this chapter, we present a technique to analyze the logical effects of physical failures in MOS circuits with the help of a multi-valued algebra. The advantage of our method over existing ones is that it is directly based on physical failures and not on any simplified fault models. The algebra was based on the results of extensive simulations using the SPICE circuit simulation program. In the following sections, the algebra for NMOS circuits is presented. It can

Table 5.1: Typical MOS Physical Failure Modes

Class	Device failures	Interconnect failures
I. Most likely	Gate to drain short. Gate to source short.	Short between diffusion lines.
II. Less likely	Drain contact open. Source contact open.	Aluminium polysilicon cross-over broken.
III. Least likely	Gate to substrate short. Floating gate.	Short between Aluminium lines.

be easily extended to CMOS circuits as well.

5.2. Development of the Model

An MOS circuit is viewed as a set of nodes connected by three-terminal devices representing transistors. The state of a node in a circuit is described by a pair <a,b>, where "a" refers to the condition of the node, and "b" refers to the logic level of the node.

Logic Levels:

The results of circuit level simulations revealed the existence of intermediate voltage levels caused by failures. This suggested the use of five logic levels for describing various voltage ranges.

0: Hard zero;

0*: Soft zero;

I: Indeterminate, near the logic threshold;

1*: Soft one;

1: Hard one.

For purpose of illustration, a typical 5 volt NMOS design [17] may be considered with a gate to source threshold voltage drop of 1.0 volt for an enhancement transistor, and -3.0 volt for a depletion transistor. For a pull-up to pull-down beta-ratio of 4 to 1, this corresponds to a logic threshold of an inverter of about 2.3 volts.

The logic level "I" represents an indeterminate level and corresponds to any voltage level between 2.0 and 2.6 volts; due to minor fluctuations in the logic threshold of different inverters receiving this input signal, it may be interpreted as a "0" or a "1".

The logic level "1" represents a "hard one"; it is always recognized as a high logic level. It represents any voltage level greater than 4.0 volts.

The logic level "1*" represents a "soft one". It is recognized to be a "1" by a logic gate but cannot drive a pass transistor fully on. It represents any voltage level in the range 2.6 to 4.0 volts.

The logic level "0" corresponds to a "hard zero"; it is always interpreted as a low logic level. It represents any voltage lower than 1.5 volts.

The logic level "0*" corresponds to a "soft zero". If it is applied to the gate of a dynamic latch, it can discharge any stored charge on the drain, if the source is grounded. It represents any voltage level between 1.5 and 2.0 volts.

Node Conditions:

Figure 5.1 demonstrates the need for labeling the state of a node as a pair. In Figure 5.1(a), given a set of logic values for the three terminals of a transistor (gate = "1", source = "0", drain = "1"), it is impossible to predict whether the "1" will be pulled down to a "0", or the "0" will be pulled up to a "1". Figure 5.1(b) shows the first case, whereas Figure 5.1(c) shows the second. Therefore, some additional information has to be provided: this is done by specifying the condition of a node. Five basic node conditions are defined.

c: Charge storage node

A node having condition "c" stores any logic value it has previously acquired. It represents a node that does not have a conducting path to VDD or ground. This models dynamic charge storage in MOS circuits.

i: Input node

All nodes representing input pads are defined to have condition "i", e.g., VDD, ground, data inputs, etc.

s: Strong driving node

A node is said to have condition "s" if it has a high conductance path to an input node. Specifically, an "s" node is connected to an input node through enhancement transistors that have their gates at a logic level "1".

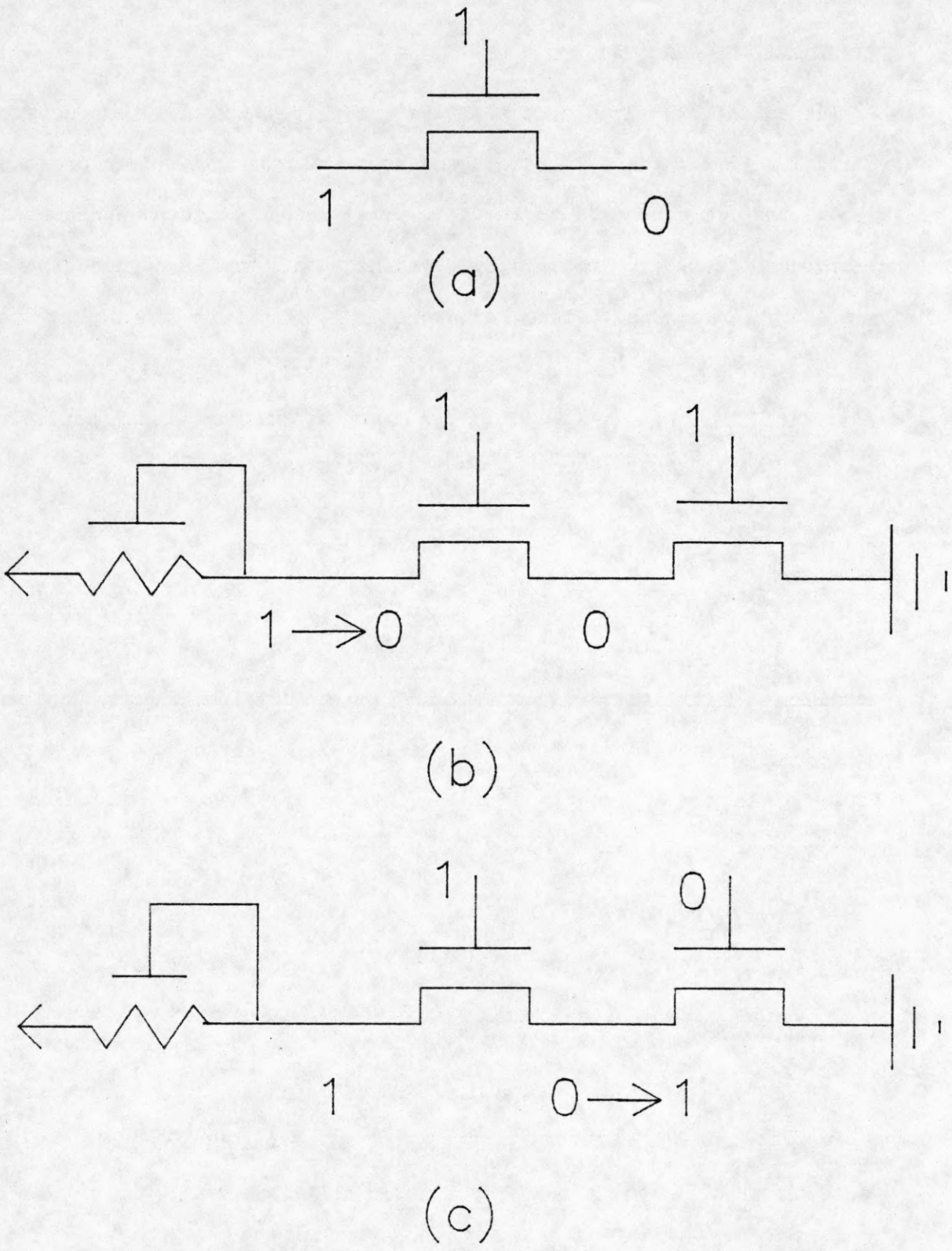


Figure 5.1: Need for Node Conditions

w: Weak driving node

The condition "w" on a node implies that it has a low conductance path to VDD (i.e., an <i,1> node), and no path to ground. It represents any node that is connected to an <i,1> node through either enhancement transistors that are not fully switched on, or through depletion transistors (acting as pull-up resistors).

f: Faulty node

This node condition represents node states that arise due to faults in an NMOS circuit.

Node States

The node conditions are dynamically assigned to various nodes depending on the states (on or off) of the transistors surrounding it. This approach is different from other switch-level approaches, such as MOSSIM [4], where a node "type" is statically assigned from the circuit description.

We have noted that we need five node conditions and five logic values to represent any "state" of a node in an NMOS circuit. However, not all combinations are allowed to exist; the reasons are derived from purely physical considerations.

A node, that has condition "i", represents an input pad such as power or ground, etc., and is allowed to have logic values of either "0" or "1". Hence, the pairs <i,1*>, <i,I>, and <i,0*> do not exist.

Since a node, having condition "s", is defined to be one that has only "fully turned on" enhancement transistors between itself and an input node, such a node cannot have a logic value of "1". This is because if a node is connected to a <i,1> node with an enhancement transistor, which has its gate at logic "1" (representing 5 volts, for example), the source of the transistor cannot be driven to more than approximately 3.5 volts, owing to the threshold voltage drop of about 1 volt (Vto = 1.0 volt) between the gate and the source of the transistor. In other words, it can only be driven to a logic level "1*", which we defined earlier. Hence the pair <s,1> cannot exist. If a node is connected to an <i,1> node via an enhancement transistor that is not fully on (i.e., its gate is driven by a logic level less than 1), then the source is driven to a logic level that is less than "1*"; such a node condition is then labeled as "w". Hence the pairs <s,I> and <s,0*> are not present.

A node is defined to have condition "w" if it has weak "pull-up" driving capability. The node state <w,0> is not allowed to exist because a node whose logic value is "0" cannot pull up any other node.

To represent states that arise from faults in a circuit, the condition "f" was introduced. It was observed from circuit simulation results, that the pairs <f,1*>, <f,I>, and <f,0*> were sufficient to cover all cases of interest.

From the above considerations, it may be concluded that out of the 25 possible node states, only 16 need to be considered to approximately describe the behavior of NMOS circuits. They are listed in Table 5.2

along with their driving characteristics. The driving strengths refer only to a relative ordering among them and not to their absolute driving strengths. The reason why the two "s" type nodes are assigned different driving strengths will be discussed later. The driving capabilities denote whether a particular node has a "pull-up", "pull-down", or "pull-self" capacity: pull-up means that it can drive a weaker node high; pull-down means that it can drive a weaker node low; pull-self means that it can drive a weaker node to its own logic value.

Tables of transistor behavior alone are not sufficient to determine the resultant behavior of an MOS circuit for a given set of inputs. This can be appreciated by referring to the example shown in Figure 5.2. The source terminals of four transistors are joined together at the node N. If the transistors were to act independently, they would each produce a different state for that node. To resolve this contradiction, a partial ordering operation has been defined in the algebra.

5.3. Partial Ordering

When several terminals are joined together at a node, they may each try to assign a different state to the node. The resultant state of a node is determined by a partial ordering among the node states, which is defined below.

With reference to Table 5.2, a partial ordering among the various node states can be defined by observing the following set of rules, which have been derived from physical considerations.

Table 5.2: Table Showing the Driving Capabilities and Relative Strengths of Various Node States

STATE	CAPABILITY	STRENGTH
<i,1 >	pull-up	5
<i,0 >	pull-down	5
<f,1*>	pull-self	4
<f,I >	pull-self	4
<f,0*>	pull-self	4
<s,0 >	pull-down	3
<s,1*>	pull-up	2
<w,1 >	pull-up	1
<w,1*>	pull-up	1
<w,I >	pull-up	1
<w,0*>	pull-up	1
<c,1 >	none	0
<c,1*>	none	0
<c,I >	none	0
<c,0*>	none	0
<c,0 >	none	0

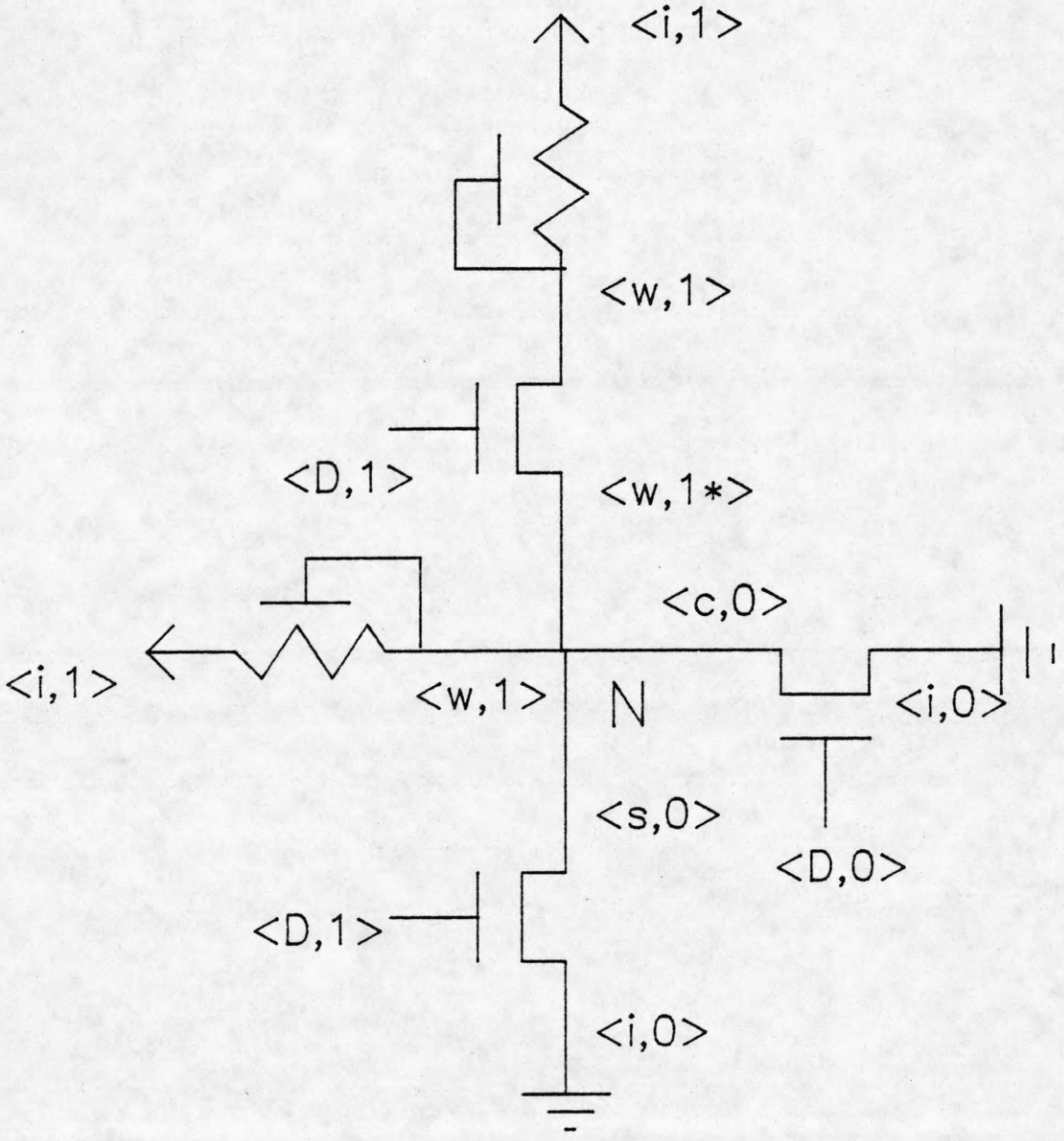


Figure 5.2: Resultant State of a Node

State A is said to be higher in the ordering than state B if and only if one of the following three conditions hold:

(1) A has higher strength than B, and A has pull-down capability.

(2) A has higher strength then B, and A has pull-up capability, and the logic value of B is less than or equal to that of A.

(3) Both A and B have the same strength, but the logic value of A is greater than that of B.

The application of the above rules results in the following partial ordering (higher from left):

<i,1>; <i,0>; <f,1*>; <f,I>; <f,0*>; <s,0>; <w,1>; <c,1>; <s,1*>; <w,1*>; <c,1*>; <w,I>; <c,I>; <w,0*>; <c,0*>; <c,0>.

If a node is assigned two or more different states from different terminals, then the resultant state for the node is the one that is the highest among them in the ordering. If M terminals are joined together at a node, each terminal i producing a node value $\langle a_i, b_i \rangle$, then the partial ordering operation P(NODE) is defined as:

$$P(NODE) = MAX\{\langle a_1, b_1 \rangle, \langle a_2, b_2 \rangle, \ldots, \langle a_M, b_M \rangle\}.$$

In Figure 5.2, for example,

$$P(N) = MAX\{\langle c,0 \rangle, \langle w,1^* \rangle, \langle w,1 \rangle, \langle s,0 \rangle\} = \langle s,0 \rangle.$$

It may be noted that the partial ordering operation between certain pairs of states would give rise to errors. For example, though <i,1> is higher than <i,0> in the ordering, a partial ordering among them would actually correspond to a physical short between the power and ground

lines giving rise to a catastrophic failure. Such failures will not be considered because their outcomes are unpredictable, and in practice such failures are easily detected because of their catastrophic nature.

5.4. Model of a Transistor

A transistor is modeled as a three terminal device whose behavior is completely specified in terms of a state table, relating the current states of the nodes at its three terminals to their next states.

The state table of a transistor can be derived from a set of rules which are related to the driving characteristics of various nodes listed in Table 5.2.

Normal enhancement transistor

The set of rules for deriving the state table of a fault free NMOS enhancement transistor is given below:

(1) The transistor is viewed as a bidirectional device between the source and the drain terminals, each of which is isolated from the gate terminal. The identity of the source and drain terminal can therefore be interchanged.

(2) The behavior of the device is independent of the condition of the node connected to the gate terminal; only the logic value of that node state is important.

(3) It is assumed that for the transistor to be switched on, there must initially exist a difference of at least one logic "step" (difference between two successive logic levels) between the gate and

the source terminals. This models the gate to source threshold voltage drop (Vto = 1.0 volts) of the enhancement transistor. The transistor can therefore alter the states of its drain and source terminals only if the above condition is satisfied.

(4) If the condition in (3) is satisfied, then the strengths of the nodes corresponding to the source and the drain are noted. The next state behavior is dictated by the node having higher strength (for example, if the nodes in question are <i,0> and <w,1>, then the behavior is dictated by <i,0>).

(5) The capability of the stronger node is next considered. If it has pull-up capability, then it can drive the weaker node to its own logic value if the latter's logic value is less than that of the former's. Otherwise, the weaker node is not affected. Similarly, if the stronger node has "pull-down" capability, then it can drive the weaker node to its own logic value only if the latter's value is greater than the former's. These changes can take place provided the gate to source threshold drop is maintained. Otherwise, the weaker node can be driven only to a value that is one "step" lower than the logic value of the gate.

(6) The next-state condition of the weaker node is dictated by the definitions of the node conditions mentioned earlier.

(7) Finally, the condition of the gate terminal is changed to "c" to model the effect of the gate capacitance.

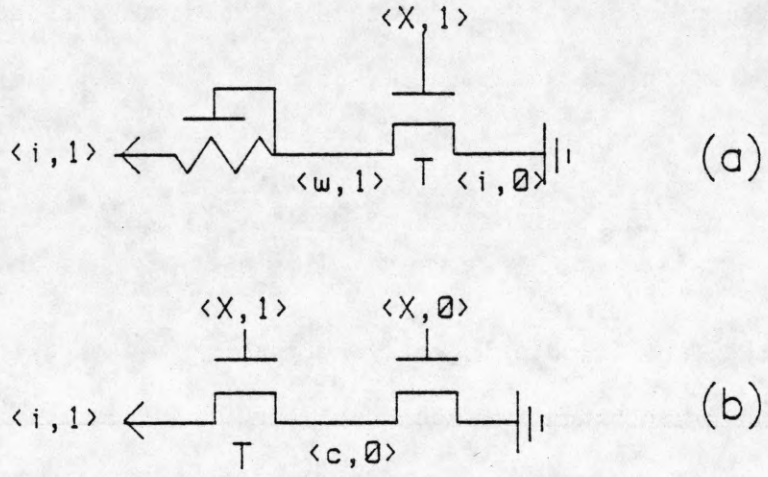
Example

Figure 5.3(a) shows an enhancement transistor T whose gate is at <X,1> ("X" means that its condition is not important), source is at <i,0> (which represents a ground node), and drain is at <w,1> (which represents a node connected to the power line through a pull-up depletion transistor). The next state behavior is determined by the stronger <i,0> node, which has pull-down capability. The logic value of the weaker node <w,1> is pulled down to "0" and its type is converted to "s", according to the definition of an <s,0> node which represents a node that is connected to an <i,0> node via fully-turned-on enhancement transistors.

Figure 5.3(b) shows the enhancement transistor T whose gate is <X,1>, source is <c,0> and drain is <i,1>. The next state behavior is determined by the stronger <i,1> node, which has pull-up capability. Hence the logic value of the weaker node, <c,0>, will be pulled up to "1*" keeping one "step" of logic level between the gate and the source to model the gate to source threshold voltage drop. The condition of the weaker node is set to "s" from the definition of an "s" node.

Faulty enhancement transistor

The behavior of an NMOS enhancement transistor under failure (the failure, for example, being a short between the gate and drain of the transistor) is described in the form of another state table. Due to the symmetry of the device, the drain and source terminals are interchangeable, so only one table is needed to describe the behavior of



PART OF STATE TABLE FOR ENH. TRANS. T :

	Current states			Next states		
	G_n	S_n	D_n	G_{n+1}	S_{n+1}	D_{n+1}
(a)	⟨X,1⟩	⟨i,0⟩	⟨w,1⟩	⟨c,1⟩	⟨i,0⟩	⟨s,0⟩
(b)	⟨X,1⟩	⟨i,1⟩	⟨c,0⟩	⟨c,1⟩	⟨i,1⟩	⟨s,1*⟩

Figure 5.3: Derivation of the State Table of an Enhancement Transistor

an NMOS enhancement device with a short between its gate and its drain or source. The set of rules are given below, assuming a short between the gate and the drain. For a short between the gate and the source of a transistor, the roles of the source and drain are reversed.

(1) If the logic value of the source is greater than "0*", then the next states of the gate and drain terminals are obtained by a partial ordering operation between those nodes. The source state is not altered.

(2) If the condition of the source is "c" (i.e., a charge storage node), then the next states of the gate and the drain nodes are obtained by taking the partial ordering between them. The state of the source node is then obtained from the behavior of the fault free enhancement transistor above.

From the results of SPICE simulation on faulty devices it was observed that three node states <f,1*>, <f,I> and <f,0*> were sufficient to model all shorted device faults. These cases will now be described.

(3) If the current states of both the gate and the drain are <w,1> (which represents a node that has a pull-up depletion transistor between itself and an <i,1> node) and the source is either <s,0> or <i,0>, then the next states of the gate and drain are <f,1*>. The source is not affected.

(4) If either the gate or drain is at <w,1> while the other is at <w,1*> (which represents a node having a depletion transistor and a fully-switched-on enhancement transistor between itself and an <i,1>

node), then with the source at <s,0> or <i,0>, the next state of the gate and drain are <f,I>. The source is again not affected.

(5) If either the gate or the drain is at <w,1>, while the other is at <c,X> (which represents a charge storage node with any arbitrary logic value), then with the source at <s,0> or <i,0>, the next states of the gate and drain are <f,0*>; the source remains unaffected.

It will be shown later that other failures such as opens in the drain and source lines, floating gates, etc. can be handled differently.

Depletion transistor

The behavior of an NMOS depletion transistor can be described by the states of the drain and source terminals only. Circuit level simulations have shown that the state of the gate terminal of a depletion transistor produces only timing changes. Observing the node states on its drain and source terminals, the next state behavior is determined by the node having greater driving strength.

(1) If the logic value of the stronger node is "X" and its condition is not "c", then the next state value of the weaker node is set to <w,X>. The state of the stronger node is not affected.

(2) Otherwise, the next states of both nodes have condition "c" with their initial logic values.

Failures, such as shorts between the gate and drain or the gate and source in a depletion transistor, do not produce any logical faults that cannot be modeled by a simple partial ordering between the relevant nodes. Other failures are handled in a way which will be described

later.

5.5. Basic Algorithm for Logic Simulation

Two primitive operations are defined in the algebra.

The first, called P(nodeid), performs a partial ordering on all the terminals joined at the node labeled "nodeid".

The second, called T(deviceid), replaces the current states of the three terminals (gate, drain and source) of the transistor, identified as "deviceid", by their next states, as obtained from the state table describing its behavior.

At the start of the simulation, all nodes that are not input nodes (i.e., of type "i") are initialized to the condition "c" (representing the charge storage capacity of those nodes), with their logic values set to "0" unless otherwise specified.

The algorithm used to obtain steady-state values of the node states, given a set of input signals, is based on an iterative procedure. The P() and T() operations are successively applied to all nodes and all transistors respectively until all the node states reach a steady state. The basic algorithm is shown in the form of a flow-chart in Figure 5.4.

Figure 5.5 shows a simple circuit, containing a 2-input NAND gate, a pass transistor, and an inverter. For the given set of inputs, the rules of the algebra are applied to obtain a steady-state solution. The three stages of iterations involved in obtaining a steady-state solution, are shown in Figures 5.6, 5.7 and 5.8.

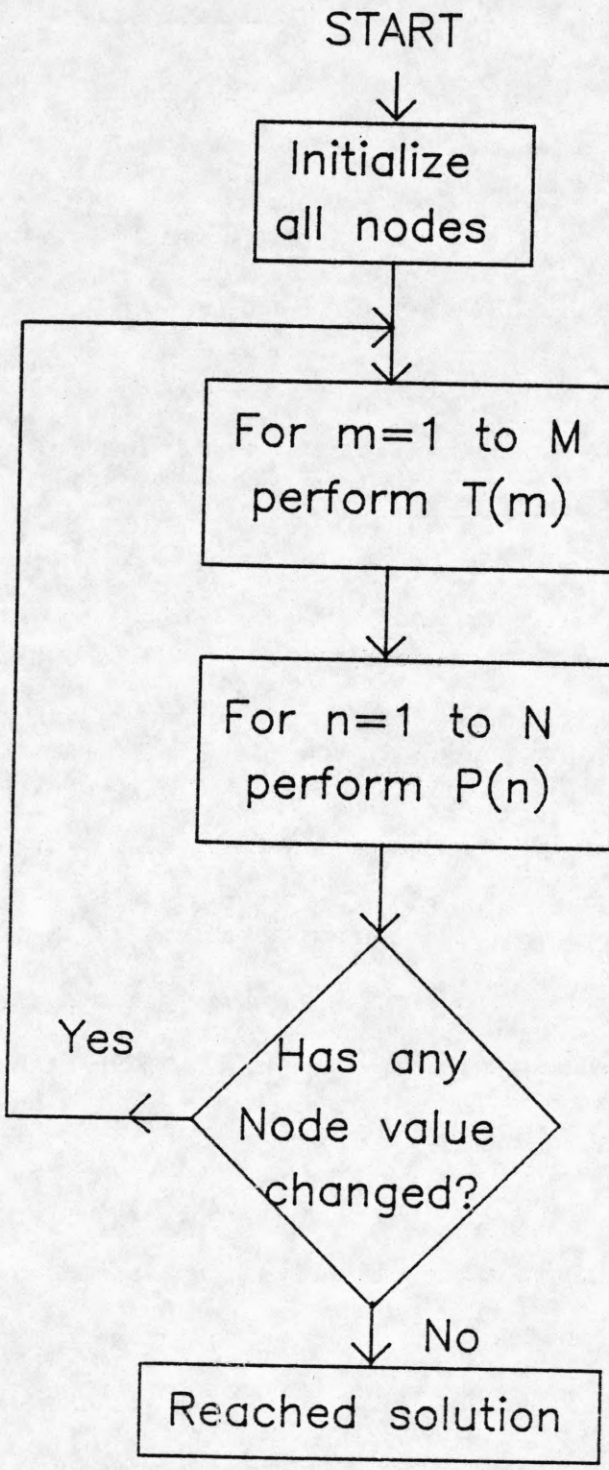
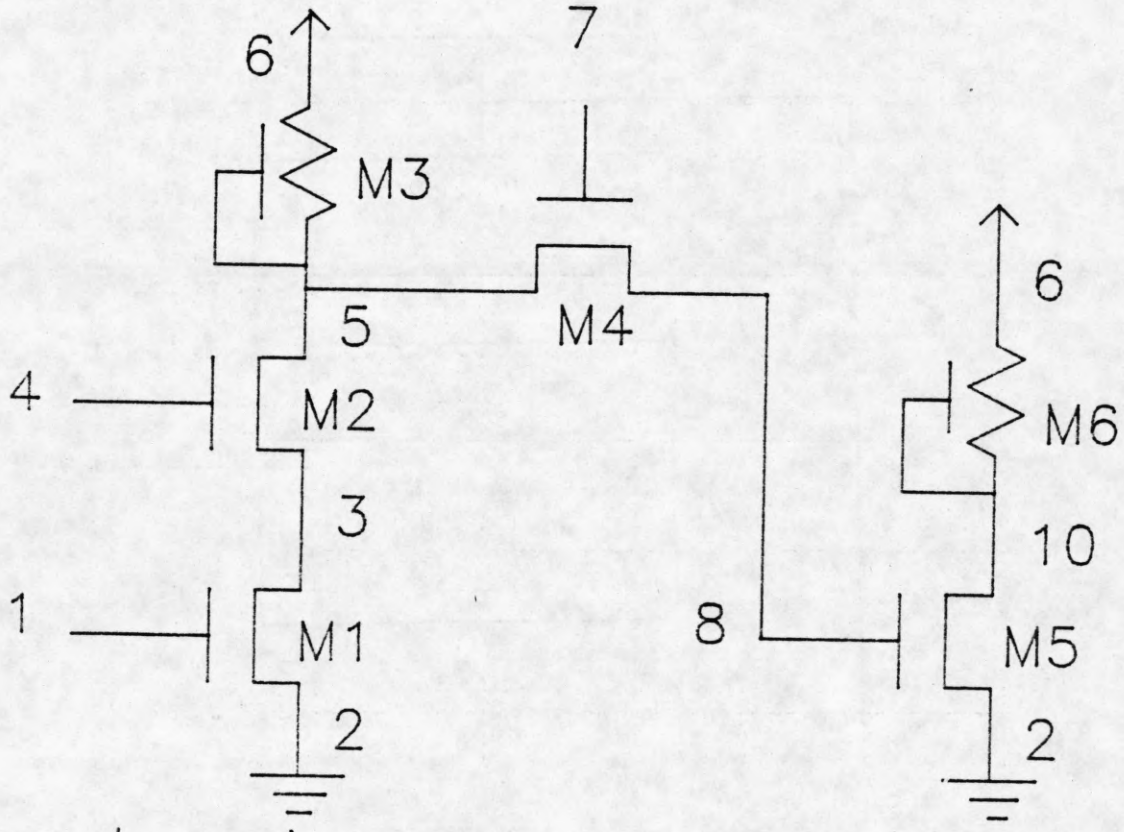


Figure 5.4: Flow-chart of Basic Algorithm



Input nodes :

1 : <i,1>, 2 : <i,0>, 4 : <i,0>

6 : <i,1>, 7 : <i,1>

Figure 5.5: Example 1

Device terminals	Current state	Next state	Input terminals
M1.G M1.S M1.D	1:<i,1> 2:<i,0> 3:<c,0>	1:<c,1> 2:<i,0> 3:<s,0>	1:<i,1> 2:<i,0> 4:<i,0> 6:<i,1> 7:<i,1>
M2.G M2.S M2.D	4:<i,0> 3:<c,0> 5:<c,0>	4:<c,0> 3:<c,0> 5:<c,0>	
M3.D M3.S	6:<i,1> 5:<c,0>	6:<i,1> 5:<w,1>	Ordering
M4.G M4.S M4.D	7:<i,1> 5:<c,0> 8:<c,0>	7:<c,1> 5:<c,0> 8:<c,0>	1:<i,1> 2:<i,0> 3:<s,0> 4:<i,0> 5:<w,1> 6:<i,1> 7:<i,1> 8:<c,0> 9:<w,1>
M5.G M5.S M5.D	8:<c,0> 2:<i,0> 9:<c,0>	8:<c,0> 2:<c,0> 9:<c,0>	
M6.D M6.S	6:<i,1> 9:<c,0>	6:<i,1> 9:<w,1>	

Figure 5.6: First Stage of Iteration

Device terminals	Current state	Next state	Input terminals
M1.G M1.S M1.D	1:⟨i,1⟩ 2:⟨i,0⟩ 3:⟨s,0⟩	1:⟨c,1⟩ 2:⟨i,0⟩ 3:⟨s,0⟩	1:⟨i,1⟩ 2:⟨i,0⟩ 4:⟨i,0⟩ 6:⟨i,1⟩ 7:⟨i,1⟩
M2.G M2.S M2.D	4:⟨i,0⟩ 3:⟨s,0⟩ 5:⟨w,1⟩	4:⟨c,0⟩ 3:⟨c,0⟩ 5:⟨c,1⟩	
M3.D M3.S	6:⟨i,1⟩ 5:⟨w,1⟩	6:⟨i,1⟩ 5:⟨w,1⟩	Ordering
M4.G M4.S M4.D	7:⟨i,1⟩ 5:⟨w,1⟩ 8:⟨c,0⟩	7:⟨c,1⟩ 5:⟨w,1⟩ 8:⟨w,1*⟩	1:⟨i,1⟩ 2:⟨i,0⟩ 3:⟨s,0⟩ 4:⟨i,0⟩ 5:⟨w,1⟩ 6:⟨i,1⟩ 7:⟨i,1⟩ 8:⟨w,1*⟩ 9:⟨w,1⟩
M5.G M5.S M5.D	8:⟨c,0⟩ 2:⟨i,0⟩ 9:⟨w,1⟩	8:⟨c,0⟩ 2:⟨c,0⟩ 9:⟨c,1⟩	
M6.D M6.S	6:⟨i,1⟩ 9:⟨w,1⟩	6:⟨i,1⟩ 9:⟨w,1⟩	

Figure 5.7: Second Stage of Iteration

Device terminals	Current state	Next state	Input terminals
M1.G	1:<i,1>	1:<c,1>	1:<i,1>
M1.S	2:<i,0>	2:<i,0>	2:<i,0>
M1.D	3:<s,0>	3:<s,0>	4:<i,0>
M2.G	4:<i,0>	4:<c,0>	6:<i,1>
M2.S	3:<s,0>	3:<c,0>	7:<i,1>
M2.D	5:<w,1>	5:<c,1>	
M3.D	6:<i,1>	6:<i,1>	Ordering
M3.S	5:<w,1>	5:<w,1>	
M4.G	7:<i,1>	7:<c,1>	1:<i,1>
M4.S	5:<w,1>	5:<w,1>	2:<i,0>
M4.D	8:<w,1*>	8:<w,1*>	3:<s,0>
M5.G	8:<w,1*>	8:<c,1*>	4:<i,0>
M5.S	2:<i,0>	2:<i,0>	5:<w,1>
M5.D	9:<w,1>	9:<s,0>	6:<i,1>
M6.D	6:<i,1>	6:<i,1>	7:<i,1>
M6.S	9:<w,1>	9:<w,1>	8:<w,1*>
			9:<s,0>

Figure 5.8: Third Stage of Iteration

Sequential behavior of a circuit, for a given sequence of input patterns, can be modeled as follows. After the steady-state values for all the nodes have been obtained for a given input pattern, each node state is initialized to <c,X> before applying the next set of inputs, where "X" denotes the logic value of the node obtained at the end of the current step.

Some Critical Remarks

It might be noted that in Table 5.2 <s,0> has been assigned a higher driving strength than <s,1*> though they are both of type "s". The reason for this lies again in the way the algebra is used to obtain a steady-state solution, which is illustrated in Figure 5.9. Consider the node N that is connected to the source terminals of two transistors T1 and T2, such that they are both on, the drain of T1 being connected to power (<i,1>), and the drain of T2 to ground (<i,0>). The transistors will independently try to produce <s,1*> and <s,0> at the node. This condition implies a high conductance path between power and ground through two "fully on" enhancement transistors. The resultant value at the node would, in practice, be indeterminate and would be represented by the faulty state <f,I> defined earlier.

The resultant of the two node states, <s,0> and <s,1*>, should give <f,I> (representing a fault condition with an indeterminate logic level), which cannot be obtained by applying the partially ordering operation mentioned earlier. To make the operation of partial ordering operation simpler, yet maintaining the accuracy of the result, we have assigned <s,0> a higher strength than <s,1*>. Then the resultant value

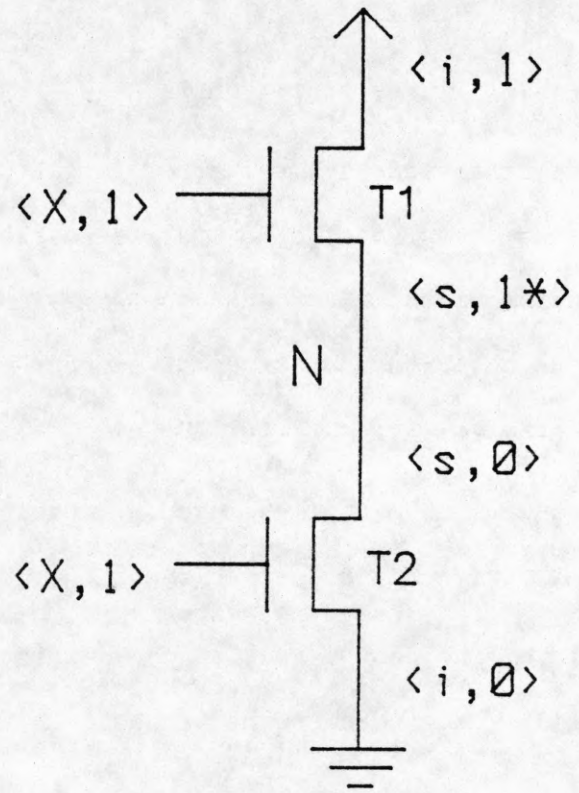


Figure 5.9: Example 2

of the node is computed to be <s,0> in an intermediate stage of the iterative procedure involving the computation of the steady-state solution. In the next stage of iteration, when T1 sees <i,1> on its drain and <s,0> on its source, it sets the source to <f,I> representing a "fault" condition. Subsequently, the application of the partial ordering operation on the node produces P(N) = MAX{<s,0>,<f,I>} = <f,I>, which gives the required result for the node.

5.6. Failures Modeled by the Algebra

All the physical failures listed in Table 5.1 can be modeled:

(1) Enhancement transistors with shorts between the gate and drain or gate and source are modeled by state tables. Owing to the bidirectional symmetry of the MOS transistor, only one faulty transistor table is required to describe both kinds of shorts: gate and drain, gate and source.

(2) A short between any two lines can be modeled by the partial ordering operation between the nodes representing those lines.

(3) An open in a line can be modeled as a splitting of the node representing the line into two disconnected nodes.

(4) Device failures, such as opens in drain or source lines or a floating gate, can be modeled by splitting the relevant node as mentioned above.

(5) The ideas developed in the derivation of the state tables for the transistors may be extended to model the effects of threshold voltage shifts. Though it would not be possible to model continuous

variations of the threshold shift, it is possible to extend the above model for quantum jumps in the threshold level. For example, in the derivation of the state table for the NMOS enhancement transistor, it was assumed that the threshold voltage drop was 1.0 volts, which corresponded to one logic "step". Models can be easily constructed for devices that have threshold voltages that correspond to two, three or four "steps".

(6) It is possible to simulate transistors that are either "stuck-open" or "stuck-closed". A "stuck-open" transistor corresponds to one whose threshold has become so high that it is in the permanently off state. Such a fault can be simulated by simply removing the device from the circuit during the simulation. A "stuck-closed" transistor corresponds to one whose threshold has shifted such that it is permanently on. Such a device can be simply simulated by removing the device, and using a partial ordering between the nodes corresponding to the drain and the source.

5.7. Limitations of the Algebra

(1) Physical failures, such as resistive shorts, and slight threshold shifts, cannot be modeled since their effects are analog in nature.

(2) No timing information is obtained.

(3) It cannot model complicated charge sharing between nodes, whose correct operation depends on the ratio of the capacitances of the relevant nodes. It is possible to extend the ideas developed above by

attaching some information regarding the relative magnitude of the capacitances of the "c" nodes to handle the charge sharing feature. To reduce the complexity of the algebra, this feature was not modeled.

5.8. Extensions to CMOS Circuits

The above algebra can be easily extended to model failures in CMOS circuits as well. The only difference is that the node condition "w" is no longer needed owing to the absence of a depletion device. Instead, two new states need to be considered: <s,0*> and <s,1>. It can be shown that 15 node states are needed in this case. The partial ordering among them is:

<i,1>; <i,0>; <f,1*>; <f,I>; <f,0*>; <s,0>; <s,1>; <s,0*>; <c,1>; <s,1*>; <c,1*>; <s,I>; <c,I>; <c,0*>; <c,0>.

State tables can be constructed for normal and faulty NMOS and PMOS transistors using rules similar to the ones indicated in this chapter.

5.9. Conclusion

Circuit level studies of the behavior of MOS circuits under physical failures have shown that the traditional fault models are inadequate for MOS technology. Unfortunately, such detailed studies are impractical for complex VLSI modules. The multi-valued algebra presented in this chapter has been developed on the basis of extensive circuit level simulations. This algebra can be used to study MOS circuits (both under failure and with no failure) with an accuracy much greater than that possible using gate level simulators, and at speeds much greater than those possible using circuit simulations. Contrary to

other approaches of using an algebra to describe MOS circuit behavior [4,14,17], this is the first instance of an algebra trying to model physical failures. Using the algebra, a simulator has been implemented. The simulator will be described in the next chapter.

CHAPTER 6

A LOGIC SIMULATOR

6.1. Introduction

In the last chapter a logical model for describing the behavior of MOS circuits was presented. Using the algebra, a simulator has been developed which could predict the behavior of MOS circuits under normal and faulty conditions. The simulator called MURPHY (which is an acronym for Mos simulation Under Realistic PHYsical failures) has been implemented in PASCAL under VAX-11/UNIX. Before discussing the details of the simulator, it is perhaps appropriate to present a short discussion on earlier attempts to build such logic simulators for MOS circuits.

6.2. Background

A logic simulator [8] has as its basis an abstract model of how digital systems function. This "logical model" describes both the structure and behavior of a system in terms of a set of primitive elements, a set of interconnections, and a set of rules for operation. For a simulator to accurately and reliably simulate a system, the logical model must reflect its actual structure and operation.

Unfortunately, the development of logic simulators has not kept pace with VLSI technology. The inadequacy stems in part from the lack of formal logic models for describing the behavior of MOS circuits. Instead, systems are designed and simulated using an ad hoc combination [22] of Boolean gate models, relay models [24], and electronic models.

The Boolean logic gate model [3,13,15] has formed the theoretical basis for logic design ever since the advent of electronic logic. In this model a system consists of a set of logic gates connected by unidirectional memoryless wires. The logic gates compute the logic functions of their input signals and transmit these values along the wires to the inputs of other wires. Each gate input has a unique signal source. Information is stored only in the feedback paths of sequential circuits. This model directly implements Boolean algebra and hence has a well-defined specification which can guide the simulator implementation.

The Boolean gate model cannot describe many of the techniques available to the logic designer, especially the MOS logic designer. MOS pass transistor networks can implement combinational logic in ways which more closely resemble relay contact networks. Dynamic memory can store information without feedback paths by exploiting the capacitances of the wires and the gates of the transistors attached to them. A logic simulator which implements only the Boolean gate model provides limited support to the MOS logic designer. Most existing logic simulators, however, extend the Boolean gate model in various ways.

Many logic simulators extend the two-valued logic of Boolean algebra with a third value to represent an unknown or undefined logic level [22]. This "X" level can indicate an uninitialized state variable, a signal held between the logic thresholds, or a signal in transition between a 0 or a 1 [5,30]. The "X" level can be handled algebraically by changing the two-valued Boolean algebra to a three-valued DeMorgan's algebra [5,30]. Thus, even with this extension many of

the undesirable mathematical properties of the Boolean gate model are preserved. To model the behavior of bus structures, some logic simulators have a fourth or "high-impedance" level [17,23]. This "H" level corresponds to the third state of tri-state logic. To simulate a bus structure, the outputs of a number of gates are connected to a common node. Typically all but one output will be at the "H" level, and the level of that output will dominate. Unlike the "X" level which can be viewed as an extension of Boolean algebra, the "H" level violates a basic principle of the Boolean model, in that a logic gate input no longer has a unique signal source.

Some simulators allow a special logic gate to represent the MOS pass transistor [27]. This logic gate models an MOS transistor as a unidirectional device with two inputs and one output. However, these simulators cannot help in cases where the bidirectional property of the device is important.

As an alternative to conventional logic simulators, a new class of logic simulators, namely switch-level simulators, have been proposed [4]. These simulators model an MOS circuit as a network of transistor "switches". They simulate many aspects of MOS circuits which cannot be expressed in the Boolean gate model, such as bidirectional pass transistors, dynamic storage and charge sharing. The nodes are assigned discrete states 0, 1 or X (for unknown), and the transistors are assigned discrete states "open", "closed" and "unknown". The operation of a network is characterized by its "target state function", which for a particular state of a network yields the logic states which the nodes

would eventually reach if all transistors were were held fixed in their initial states.

Numerous attempts have been made to model the effects of physical failures on digital circuits at the logical level. A commonly used fault model is the "stuck-line" model. This model assumes that physical failures inside logic gates can be modeled as lines in the input or output of the logic gates to be permanently "stuck" at 0 or 1. The stuck-line fault model is quite attractive because of its simplicity. It can be easily handled by Boolean algebra. It has, therefore, provided a basis for many conventional fault simulators that were based on the Boolean gate model discussed earlier.

However, it has been widely noted that this simple fault model does not cover a large class of physical failures that are observed in MOS circuits [28,29]. Models such as the "stuck-open" and "stuck-closed" fault models have been proposed. They can cover some of the physical failures that were not covered by the earlier fault models. This fault model views the MOS transistor as a switch and assumes that under failure the MOS transistor can behave as if it were permanently on or off. At the gate level, this fault can be modeled by the addition of pseudo-gates by a gate level fault simulator. Such fault simulators are cumbersome; besides these pseudo-gates have no physical interpretation. Also, from the results of our study of physical failures on MOS circuits using circuit level simulation, we have noted that the there are many failures which cannot be modeled by any of the above fault models.

With the kinds of failures that we want to study, it is not possible to extend the ideas of a switch level model of an MOS circuit to build a fault simulator. This is due to the fundamental assumption in the switch level model that the gate of a transistor is isolated from the source and the drain. Therefore, they cannot simulate failures such as a short between the gate and source or drain of a transistor, a failure which is extremely likely to occur, as mentioned earlier in the thesis. Hence, simple switch level simulations will not suffice. We have,therefore, developed a simulator which is based on the multi-valued algebra described in Chapter 5. This simulator can simulate faults which have direct correspondence with physical failures that are typically observed in the field.

6.3. Introduction to MURPHY

In the following sections, a brief description of the simulator MURPHY will be presented. The various steps involved in the design of the final simulator will be discussed. Starting from a crude basic simulation algorithm, various strategies of improving the performance of the simulator will be indicated. Their relative merits and demerits will be discussed as well.

In the discussion of the simulator, we shall refer to the "state" of a node as representing the pair <a,b>, mentioned in Chapter 5. The data structures used in the internal representation of an NMOS circuit are described below.

Data Structures

Four kinds of data objects are defined:

(1) The object DEVICE_TERMINAL is used to represent terminals of transistors, namely, the gate, the drain, and the source.

(2) The object INPUT_TERMINAL represents terminals such as power and ground.

(3) The object TRANSISTOR corresponds to transistors in a real circuit.

(4) The object NODE corresponds to circuit nodes where terminals are connected.

The junction of several terminals at a circuit node (represented by a NODE) is internally represented as a linked list of INPUT_TERMINALs and DEVICE_TERMINALs. The first element in the list is pointed to by a NODE. There are as many linked lists in the internal representation of a circuit as there are nodes.

A DEVICE_TERMINAL contains the following information: the type of terminal it represents (e.g., gate, source, etc.); the type and identity of the device to which the terminal belongs; the "state" with which it is to be initialized at the beginning of the simulation; its current "state"; a pointer to the next member in the linked list corresponding to the circuit node where this terminal is connected.

The structure of an INPUT_TERMINAL specifies: the identity of the input; its current value; a pointer to the next member of the linked list corresponding to the circuit node it is connected to.

A TRANSISTOR is represented by a structure which points to three DEVICE_TERMINALs, representing its gate, drain and source terminals.

The structure of a NODE contains the following information: the resultant "state" of the node from the previous iteration; a flag which indicates how the resultant "state" of the node has changed during the present iteration. A pointer to the head of a linked list of INPUT_TERMINALs and DEVICE_TERMINALs connected to this NODE.

Primitive Operations

Two basic operations are defined:

P(n) performs a partial ordering on the states of all members of the linked list corresponding to NODE "n" to compute the resultant state; it then replaces the old states by the resultant state. The current and previous states of the NODE are compared, and a flag is set accordingly.

T(m) performs a table look-up of the appropriate state table corresponding to TRANSISTOR "m", and replaces the current states of the DEVICE_TERMINALs corresponding to its gate, source and drain by their next states.

6.4. Global Simulation Algorithm

The simulation begins by initializing the states of all NODEs to <c,0>, unless otherwise specified. The flags of all the NODEs are reset. The simulation involves an iterative procedure. Each step involves the application of the "T" operation on all the transistors, followed by the

"P" operation on all the NODEs, until the resultant states of all the NODEs remain unchanged during two successive iterations.

Roll-back

However, such a simple procedure will not always produce the correct result owing to certain "race" conditions internal to the simulator. These races occur due to the assignment of false high logic values to certain nodes during some intermediate steps, which cause undesirable switching on of some transistors. To counteract such races, certain roll-back mechanisms are incorporated.

Once it is detected that the logic value of the current state of a NODE is less than that of the previous state, a procedure "Roll-back" is called.

This procedure searches through the linked list of DEVICE_TERMINALs that are connected to the target NODE for terminals that represent gates of enhancement TRANSISTORs.

For each such device D, the TRANSISTORs and NODEs that are "connected" to the source or the drain of device D via "on" enhancement transistors (i.e., those that have logic values of either I, 1*, or 1 at their gates) are pushed into a stack.

The NODEs on the stack are set to their initial states; a steady-state solution is obtained locally for the TRANSISTORs and NODEs on the stack, using the general iterative procedure. While obtaining a local solution, if any of the final NODE states changes in such a manner as to result in a decrease in its logic value, the roll-back procedure is

called recursively.

The modified algorithm is given below:

```
PROCEDURE MURPHY_1;

BEGIN
  Initialize;
  done := FALSE;
  WHILE (NOT done) DO
    BEGIN
      done := TRUE;
      FOR m := 1 TO numberoftransistors  DO
          T(m);
      FOR n := 1 TO numberofnodes DO
          P(n);
      FOR n := 1 TO numberofnodes DO
        BEGIN
          IF (node[n].currentstate) <> (node[n].previousstate)
             THEN done := FALSE;
          IF (node[n].currlogicval) < (node[n].prevlogicval)
             THEN Roll-back(n);
        END;
    END;
END;
```

It is possible to simulate sequential behavior of NMOS circuits using dynamic charge storage. After obtaining the steady-state solution for a particular input combination, the state <D,X> of every NODE "n" is set to <c,X>, where "X" is the logic value, before beginning the computations for the next input combination.

For example, if a NODE attains the state <w,1*> at the end of a computation step, the states of all the DEVICE_TERMINALs in the linked list (corresponding to the NODE) are set to <c,1*> before proceeding with the computations for the next set of inputs.

Remarks

The first version of the simulator, MURPHY-1, was implemented in PASCAL under UNIX on a VAX/11 computer. It was found to be reasonably accurate with regard to predicting the behavior of faulty NMOS circuits. It was found to be about 100 times faster than SPICE while simulating circuits containing about 20 nodes and 40 transistors. Some typical performance figures are shown in Table 6.1.

The major drawback of the simulator described above was that it attempted to obtain solutions for the circuit as a whole. This resulted in a considerable amount of unnecessary computation.

For example, to simulate a string of 100 inverters (which consists of about 100 nodes and 200 transistors), the simulator required about 100 iterations to converge to a steady-state solution. Each iteration involved the application of the "P" operation on all 100 nodes and the "T" operation on all 200 transistors: this added up to about 10000 "P" and 20000 "T" operations which clearly required a considerable amount of time.

If instead, it were possible to partition the circuit into smaller groups consisting of a few levels per group, then the simulation of a large circuit could be performed by dividing the task into smaller tasks. Solution for each group (in this case, an inverter) would take only 2 iterations. This technique of simulation of the circuit would therefore produce a substantial reduction in the amount of computation. Such a partitioned approach will be described next.

Table 6.1: Comparison of SPICE and MURPHY_1

Circuit	#Nodes	#Devices	SPICE	MURPHY_1		Improvement factor
			simulation time (sec)	setup time (sec)	simulation time (sec)	
Inverter string	23	42	85.0	0.200	0.416	138
Delay register	10	12	22.0	0.033	0.218	88
NOR type Decoder	13	22	43.0	0.117	0.217	129
NOR-NOR P.L.A.	25	36	77.0	0.150	0.530	113

6.5. Partitioning Algorithm

In order to partition a large NMOS circuit into smaller sub-circuits, the circuit is internally viewed as a graph. The vertices in the graph correspond to circuit nodes.

Two types of vertices are defined in the graph, represented differently by triangles (type "T") and boxes (type "B"). "T" vertices represent those circuit nodes that are connected to an external input terminal, such as power, ground or data inputs. All other vertices are of type "B".

Two types of edges are defined in the graph, weak and strong. Strong edges (type "S") are undirected and are represented by continuous lines; weak edges (type "W") are directed and represented by dotted lines.

A fault-free enhancement transistor is represented by three edges:

(1) a strong undirected edge ("S") between the vertices corresponding to the drain and source terminals;

(2) a weak directed edge ("W") from the gate vertex to the drain vertex;

(3) a weak directed edge ("W") from the gate vertex to the source vertex;

A depletion transistor is represented by a single strong undirected edge ("S") between the drain and the source vertices. Since there is no effect of gate terminal on the logical behavior of a depletion transistor (except timing effects, which are not modeled), the gate

connection is ignored.

To model a physical "short" between two circuit nodes, an undirected strong edge is introduced between the vertices representing those nodes.

For example, a faulty enhancement transistor with a short between the gate and the source is represented by two strong edges and one weak edge:

(1) a strong undirected edge ("S") between the drain and the source vertices;

(2) a strong undirected edge ("S") between the gate and the source vertices;

(3) a weak directed edge ("W") from the gate vertex to the drain vertex.

A faulty enhancement transistor with a short between the gate and the drain is represented in a similar manner by two strong edges and one weak edges.

The label of an edge refers to the identity of the transistor to which the edge belongs.

Figure 6.1 shows a circuit whose graphical representation is shown in Figure 6.2.

Using the above definitions, a large circuit is partitioned into groups. The object GROUP contains the following information: the group identifier; headers to three linked lists: TRANSISTORS, EXTERNAL_NODES and INTERNAL_NODES. The algorithm is described below:

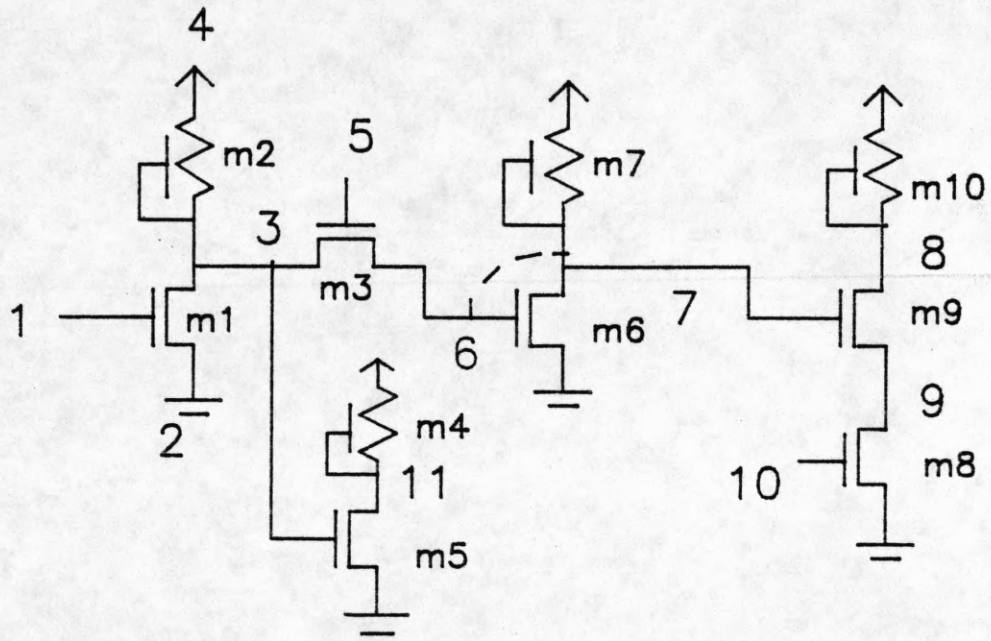


Figure 6.1: Example Circuit for Partitioning

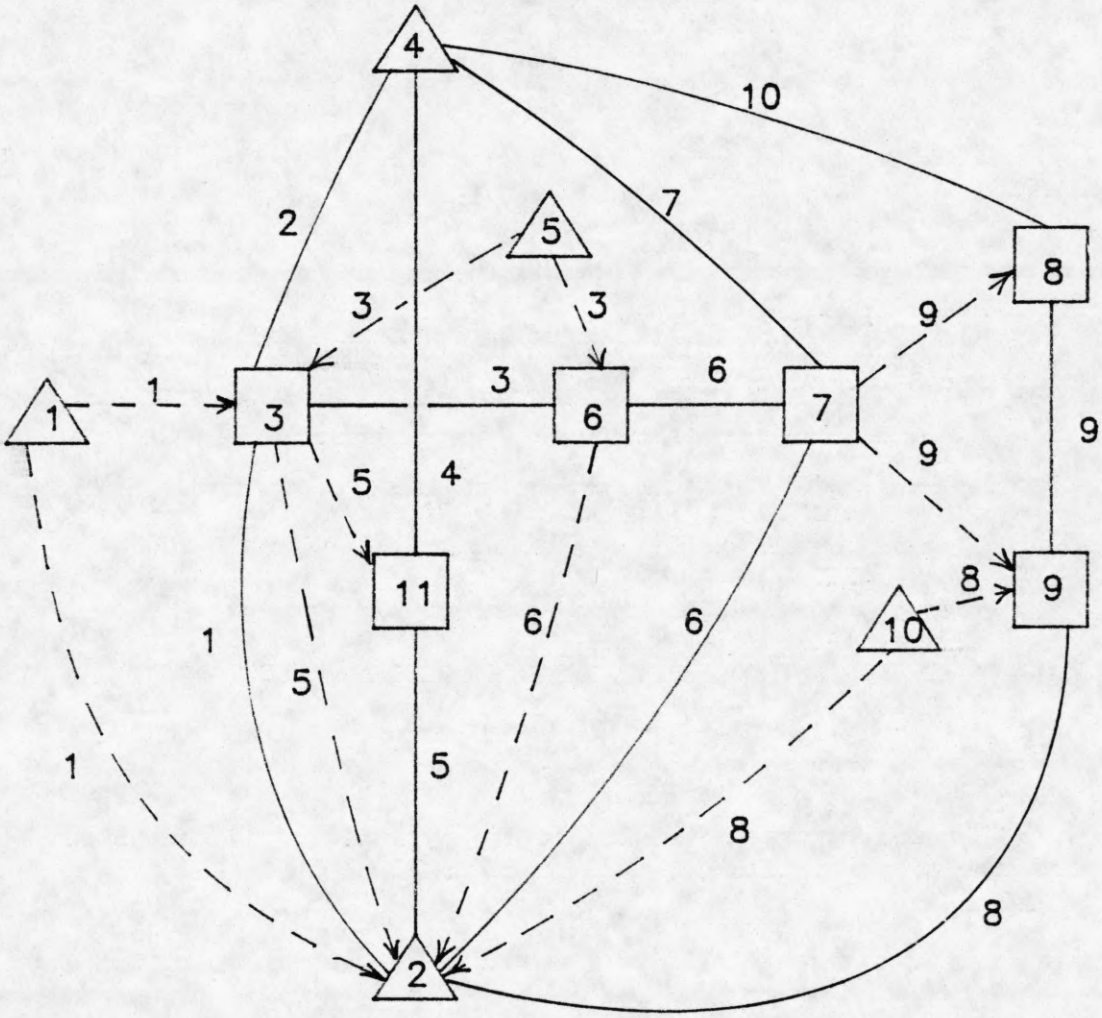


Figure 6.2: Graphical Reprentation of Example Circuit

```
PROCEDURE PARTITION;

BEGIN
  count := 0;   (* count of number of groups formed *)
  FOR each vertex "v" in the graph DO
     visited(v) := 0;
  FOR each vertex "v" in the graph DO BEGIN
    IF (visited(v) = 0) AND (vertex "v" is of type "B")
       THEN BEGIN
          count := count + 1;
          create new GROUP "count";
          FORMGROUP (v,count);
          CLEANUP(count);
          output GROUP "count";
       END;
  END;
END;



PROCEDURE FORMGROUP (v,g);
(* form lists of TRANSISTORS, EXTERNAL_NODES, INTERNAL_NODES
   for GROUP "g" starting from vertex "v"                    *)

BEGIN
  visited(v) := 1;
  add "v" to INTERNAL_NODE_LIST of GROUP "g";
  FOR each vertex "u" linked to "v" by "S" edges DO BEGIN
    b := label of "S" edge;
    add "b" to TRANSISTOR_LIST for group "g";
    IF (vertex "u" is of type "T")
      THEN add "u" to EXTERNAL_NODE_LIST for GROUP "g"
      ELSE
        IF visited(u) = 0
          THEN FORMGROUP (u,g);
  END;
  FOR each vertex "p" linked to "v" by incident "W" edges DO
    add "p" to EXTERNAL_NODE_LIST for GROUP "g";
END;



PROCEDURE CLEANUP(g);

BEGIN
  FOR each element of the INTERNAL_NODE_LIST DO
     IF that element appears in the EXTERNAL_NODE_LIST
        THEN delete element from EXTERNAL_NODE_LIST;
END;
```

Figure 6.3 lists the various groups in the graph after the application of the procedure PARTITION on the graph shown in Figure 6.2.

Complexity of the Partitioning Algorithm

Suppose the circuit has N nodes and P transistors. Then the graph has N vertices and E edges, where

MAXIMUM(E) = 3 * P,

for the case where all transistors are of the enhancement type.

However, the number of type "B" vertices is equal to

M = N - number of "T" vertices.

The algorithm PARTITION performs a depth-first search of the vertices in the graph. The execution time of the procedure FORMGROUP is bounded by the number of edges incident at the vertex "v" ignoring the recursive call. In the algorithm PARTITION, the procedure FORMGROUP is called at most M times, once for every vertex of type "B".

Hence the time complexity of the algorithm is

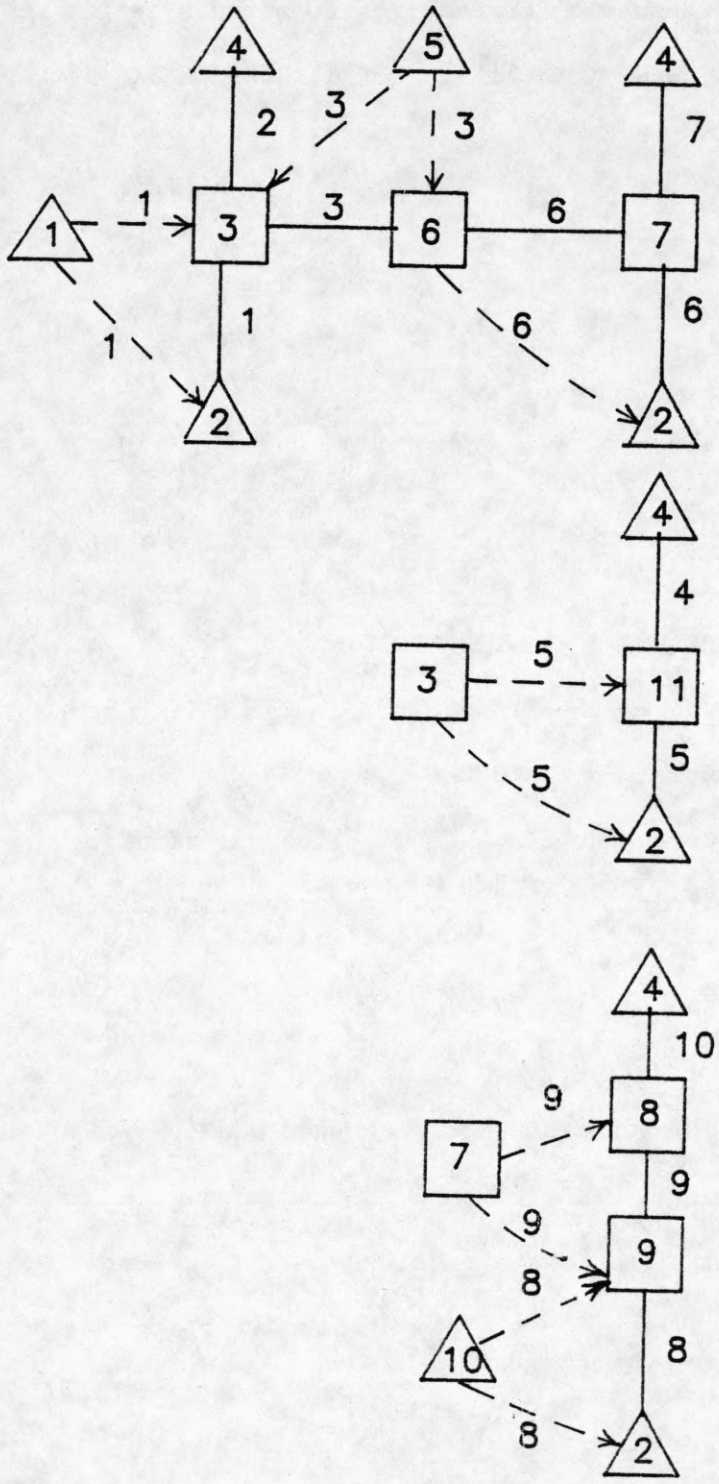
Time = O(MAXIMUM(M,E)),

since each type "B" vertex and each edge is visited at most once.

Hence the partitioning time is linear with the number of nodes or transistors in the circuit.

6.6. Simulation of a Group

Once a circuit has been partitioned into smaller groups, the procedure for simulation of the whole circuit consists of individually



GROUP 1

Trans. = 1,2,3,6,7
Int. nodes = 3,6,7
Ext. nodes = 1,2,4,5

GROUP 2

Trans. = 4,5

Int. nodes = 11
Ext. nodes = 2,3,4

GROUP 3

Trans. = 8,9,10
Int. nodes = 8,9
Ext. nodes = 2,4,7,10

Figure 6.3: Groups in the Graph

obtaining solutions for the groups. For each group, given a set of node states for its EXTERNAL nodes, the states of its INTERNAL nodes are determined by the procedure LOCAL. The algorithm for obtaining a local solution for a particular group is given below.

```
PROCEDURE LOCAL(group);

BEGIN
  FOR all INTERNAL nodes "n" in the GROUP DO
    Initialize node "n" to <c,X>;
  done := FALSE;
  WHILE (NOT done) DO
    BEGIN
      done := TRUE;
      FOR all TRANSISTORS "m" in the GROUP DO
          T(m);
      FOR all INTERNAL and EXTERNAL nodes "n" in GROUP DO
          P(n);
      FOR all INTERNAL nodes "n" in GROUP DO
        BEGIN
          IF (node[n].currentstate) <> (node[n].previousstate)
             THEN done := FALSE;
          IF (node[n].currlogicval) < (node[n].prevlogicval)
             THEN Roll-back(n);
        END;
    END;
END;
```

The complexity of the solution of the procedure LOCAL is proportional to the number of TRANSISTORS or NODES (whichever is larger) times the number of iterations required to converge. The number of iterations required to converge for a group is bounded by the longest "path" between any two nodes in the group. A "path" is any cycle-free sequence of nodes connected by "S" edges in the graph. For practical NMOS circuits there are less than 4 levels of pass transistors in any chain. Hence, the number of iterations in a group is bounded by a

constant. The time required for simulation of a group is therefore linear with the number of transistors or nodes in the group.

The order in which local solutions are obtained for various groups will be described next.

6.7. Event Driven Simulation Algorithm

In this approach an event list of groups requiring at least one more simulation pass is maintained dynamically. The event list is in the form of a first-in-first-out (FIFO) queue. A scheduler places groups at the front of the queue when it determines that a group is a candidate for another simulation pass. A group is a candidate for a simulation pass if:

(1) the group has not been solved at all; so, initially all groups are placed on the queue.

(2) During the global simulation a group is placed on the queue only if the logic value of one of its EXTERNAL_NODES has been altered from the value assumed in the previous simulation pass for that group.

The central process performing the simulation of the circuit removes groups from the head of the event list and performs the procedure LOCAL on that group. The simulation ends when the event list is empty. The algorithm is given below.

```
PROCEDURE MURPHY_2;

BEGIN
  FOR all GROUPS "g" DO
    put_on_event_list(group[g]);

  WHILE event_list not empty DO BEGIN
    g := remove next group from event_list;
    LOCAL(g);
    FOR each node "n" in INTERNAL_NODE_LIST DO
      FOR each GROUP "f" whose EXTERNAL_NODE_LIST contains "n" DO
        IF logic_value of EXTERNAL_NODE "n" has changed
           THEN put_on_event_list(group[h]);
  END;
END;
```

The above algorithm simulates large NMOS circuits much faster than the earlier algorithm MURPHY_1. However, while solving for circuits involving feedback and fan-out it is quite inefficient. This can be understood from Figure 6.4, which shows a directed graph representation of an NMOS circuit involving feedback. Each box represents a group. The arcs represent dependencies of the groups. The arcs (representing circuit nodes) are directed from the group of which it is an INTERNAL_NODE to the groups of which it is an EXTERNAL_NODE.

A very simple example has been considered to illustrate the point that the algorithm MURPHY_2 is inefficient. After solving for group 1, the algorithm arbitrarily assigns the starting state of L_3 to <c,0> and solves for group 2. Now since L_4 and L_{10} have been solved, the groups 3 and 8 are placed on the queue. Next when group 3 is solved, group 4 is put on the queue. Subsequently, when group 8 is solved, groups 9 and 13 are put on the queue. Similarly when group 4 is solved, group 17 is placed on the queue. But before group 17 can be solved, groups 9 and 13 are solved. After solving for group 17, if it is determined that the

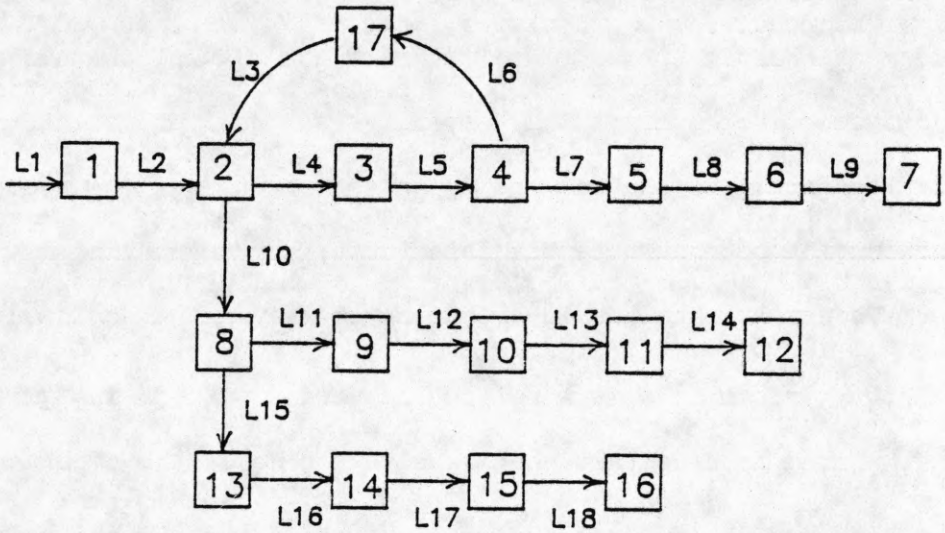


Figure 6.4: Dependency Graph for Groups

logic value of L_3 has changed, the entire process has to be repeated. Every time an iteration around the loop is required, all the groups in the circuit that fan out from any point in the feedback loop have to be solved.

This algorithm is therefore quite inefficient. A third algorithm based on a static analysis of the dependencies will next be presented.

6.8. Static Labeling Algorithm

The central idea of the algorithm to be discussed is a static labeling of the different groups prior to any simulation passes. Groups are assigned level numbers which specify the order in which the groups should be simulated keeping the dependencies in mind. A group having a higher level number must be simulated after a group having a lower level number. Groups at the same level number can be simulated in any order.

The algorithm used in labeling the various groups is given below. The algorithm assumes that there are no cycles in the dependency graph, a constraint which is satisfied for combinational circuits. Circuits involving feedback are handled in a way which is described later.

```
PROCEDURE LABELING_1;

BEGIN
  count := 0;
  FOR g := 1 TO numgroup DO
    GROUP[g].level := 0;
  FOR n := 1 TO numnode DO
    IF NODE[n].type = "T"
      THEN NODE[n].level := 1
      ELSE NODE[n].level := 0;
  REPEAT
    changed := false;
    FOR g := 1 TO numgroup DO BEGIN
      IF GROUP[g].level = 0
         THEN BEGIN
            IF all EXTERNAL_NODES have levels > 0
              THEN BEGIN
                 GROUP[g].level := MAX (EXTERNAL_NODE.levels);
                 add GROUP "g" on list of groups at level "MAX";
                 count := count + 1;
                 changed := true;
                 FOR all NODES "n" on INTERNAL_NODE_LIST of "g" DO
                    NODE[n].level := MAX + 1;
              END;
         END;
    END;
  UNTIL NOT changed;
END;
```

In the procedure LABELING_1 the "FOR g" loop is executed G times (where G is the number of groups to be labeled). The outer REPEAT loop is then executed at most G times if we assume that in the worst case only one group can be successfully labeled for an iteration of the outer loop (if no group can be labeled during an outer loop iteration then the procedure ends). The worst case time complexity of the algorithm is therefore given by

Time = $O(G^2)$, where G is the number of groups.

After all the groups have been labeled and placed on linked lists at various levels LEVEL(L), the groups are solved by the algorithm given below. The call to the procedure LOCAL is to the same procedure described in the earlier section.

```
PROCEDURE MURPHY_3;

BEGIN
  Initialize;
  FOR L := 1 TO maximum_number_of_levels DO BEGIN
     WHILE LEVEL(L) not empty DO BEGIN
        take next group g from LEVEL(L);
        LOCAL(g);
     END;
  END;
END;
```

The above algorithm is much more efficient than MURPHY_2. For combinational circuits with fanout, it requires only one pass to converge to the solution.

It was noted earlier that the above labeling algorithm fails for dependency graphs of groups having cycles. This case is handled by detecting strongly connected components in a directed graph [1]. The algorithm is given below.

```
Input: Directed graph G = (V,E) : V is set of vertices,
       E is set of edges.

Output: A list of strongly connected components of G.

ALGORITHM STRONG_COMP(V,E);

BEGIN
  count := 1;
  comp := 0;
  FOR all "v" in V DO
    mark "v" new;
  Initialize STACK to empty;
  WHILE there exists a vertex "v" marked new DO BEGIN
      comp := comp + 1;
      SEARCH(v,comp);
  END;
END;



PROCEDURE SEARCH(v,comp);

BEGIN
  mark "v" old;
  dfnumber[v] := count;
  count := count + 1;
  lowlink[v] := dfnumber[v];
  push "v" on STACK;
  FOR each vertex "w" linked to "v" DO
    IF "w" is marked new
      THEN BEGIN
        SEARCH(w,comp);
        lowlink[v] := MIN (lowlink[v],lowlink[w]);
      END
      ELSE
        IF dfnumber[w] < dfnumber[v] AND "w" is on STACK
          THEN lowlink[v] := MIN (dfnumber[w],lowlink[v]);
  IF lowlink[v] = dfnumber[v]
     THEN
        REPEAT
          pop "x" from top of STACK;
          output "x" as belonging to component "comp";
        UNTIL x = v;
END;
```

It is shown in [1] that the complexity of the above algorithm in time is O((MAX(V,E)).

This algorithm was used to identify groups belonging to a cycle in the dependency graph (of groups). The algorithm in MURPHY_3 was therefore slightly modified to include the capability of simulating circuits with feedback as well.

The modification is that after partitioning the circuit into groups, a dependency graph is constructed (there is no great overhead involved in doing this. The procedure PARTITION can keep track of the dependencies). This dependency graph is analyzed by the algorithm STRONG_COMP which identifies the strongly connected components in the directed graph. Subsequently, the components are searched for the number of groups in each component. A component having more than one group corresponds to a cycle. A cycle-free component is kept unaltered. For a component containing a cycle, all groups in a cycle are collapsed into one large group.

Finally, the procedure LABELING_1 is called on these modified groups. The procedure MURPHY_3 then remains the same.

Comment on the complexity

By analyzing the procedures involved in the simulation of a large circuit using MURPHY_3, it is seen that all procedures except LABELING_1 can be executed in times that are linear with the number of nodes or transistors. Though from the computational theory viewpoint, the resultant time complexity is still of the order of the square of the

number of strongly connected components in the dependence graph, for simulations of real circuits it is still quite efficient.

This is because all this overhead goes in before simulating the circuit for any test inputs. Once all the overhead has been performed, the circuit can be simulated very fast since the simulation time is linear with the number of nodes or transistors.

For circuits involving feedback, the simulation time is governed by the time required for simulating large collapsed groups. If the circuit is such that the collapsing of components with cycles leads to groups which are as complex as the original circuit itself, then the performance during actual simulation becomes the same as MURPHY_1. Hence, a fourth algorithm will be described next which is a combination of the algorithms MURPHY_2 and MURPHY-3.

6.9. Hybrid Algorithm

It was decided that a mixture of static labeling and event-driven simulation would probably be the best. Here, instead of collapsing the groups in a cycle into a larger group, and then calling procedure LABELING_1 to perform a static labeling of the groups, the groups in a cycle are all assigned a special level. The groups on such special levels are simulated in an event-driven fashion. Groups that do not belong to any cycle are simulated as in MURPHY_3, according to their static levels.

The algorithm is given below:

```
PROCEDURE LABELING_2;

BEGIN
  count := 0;
  levelnum := 0;
  FOR c := 1 TO numcomp DO
    COMPONENT[g].level := 0;
  FOR n := 1 TO numnode DO
    IF NODE[n].type = "T"
      THEN NODE[n].level := 1
      ELSE NODE[n].level := 0;
 REPEAT
  REPEAT
    changed := false;
    FOR c := 1 TO numcomp DO BEGIN
      IF COMPONENT[c].level = 0
       THEN
         IF COMPONENT[c] not cycle
           THEN BEGIN
            g := group in COMPONENT[c];
            IF all EXTERNAL_NODES of "g" have levels > 0
              THEN BEGIN
                 GROUP[g].level := MAX (EXTERNAL_NODE.levels);
                 if levelnum < GROUP[g].level
                     THEN levelnum := GROUP[g].level;
                 add GROUP "g" on list of groups at level "MAX";
                 count := count + 1;
                 changed := true;
                 FOR all NODES "n" on INTERNAL_NODE_LIST of "g" DO
                    NODE[n].level := MAX + 1;
              END;
           END;
    END;
  UNTIL NOT changed;
    IF NOT changed
       THEN BEGIN
          search for a component with a cycle;
          identify groups in the cycle;
          IF all EXTERNAL_NODES of such groups EITHER
                 1) have level > 0,  OR
                 2) are INTERNAL_NODES of groups in cycle
           THEN BEGIN
             changed := true;
             levelnum := levelnum + 1;
             place all GROUPS in cycle at special level "levelnum";
             FOR all INTERNAL_NODES n belonging to GROUPS DO
               node[n].level := levelnum + 1;
             count := count + number_of_groups_in_component;
             END;
          END;
       END;
```

```
  UNTIL count >= numgroup;
END;
```

The overall algorithm for simulating a circuit is then given below:

```
PROCEDURE MURPHY_4;

BEGIN
   Initialize;
   FOR L := 1 to levelnum DO BEGIN
      if level[L] is a special level (* cycle present *)
         THEN event-driven(level[L])
         ELSE
           WHILE level[L] list not empty DO BEGIN
              g := take next group from level L;
              LOCAL(g);
           END;
   END;
END;
```

Solving for feedback loops can take more than one pass locally (among the groups that are actually involved in the feedback loops) using an event-driven approach, but globally it still requires one pass to obtain a steady-state solution.

6.10. Conclusion

All four simulation algorithms have been implemented in PASCAL under UNIX on a VAX/11 computer. Compared to the VAX/SPICE circuit simulation program, all are about two orders of magnitude faster while simulating circuits which have about 40 transistors.

Some preliminary studies have shown that the partitioning approach significantly reduces the simulation time. It is expected that for really large circuits the improvements will be particularly noticeable.

The advantage of using the simulator MURPHY is that it can simulate the behavior of NMOS circuits (with or without failure) with an accuracy much greater than that of gate-level or switch-level simulators, and with speeds much faster than those possible using a circuit level simulator. This is probably the first attempt to model physical failures directly in a logic simulator.

MURPHY has been found to correctly simulate static and dynamic logic circuits such as latches and flip-flops. However, it fails to converge for circuits, such as a Ring Oscillator, that have regenerative feedback in them. It must be noted that no existing logic simulator converges for such circuits.

It does not simulate the behavior of charge sharing in NMOS circuits, since there is no way of indicating the relative magnitudes of the capacitance of the nodes. When the simulator encounters such a condition, such as an "on" transistor connected between a <c,0> and a <c,1*> node, it sets both nodes to <c,1*>, that is, their logic value is set to the larger of the two values.

The simulator MURPHY can simulate NMOS circuits only. However, by extending the algebra to CMOS circuits, as indicated in Chapter 5, it is possible to use the simulator for CMOS circuits.

The simulator can be used as a useful tool for developing functional level fault models for complex VLSI modules. It can also be used as a logic design verification tool by interfacing it to a component layout extraction program to verify the correctness of the layout of a VLSI module at the logical level.

CHAPTER 7

CONCLUSION

Tests for detecting faults in integrated logic circuits must be specifically designed to recognize failure-mode dependence on circuit configuration, processing parameters, and technology (TTL, STL, CMOS, NMOS, etc.). Traditionally, a gap exists between the logic designer, the circuit designer, and the process engineer.

The process engineer is only concerned with better ways of fabricating smaller and faster devices on a chip. Though he has a good idea of the different physical failure modes at the device level, e.g., electro-migration, whisker formation, and mask misalignment, he is not concerned with the effects these failures have on logic circuits.

The circuit designer is traditionally involved with designs to realize a given logic function in a way so as to produce the maximum speed and minimal area. While designing these circuits, he rarely considers the possibility of failures on the chip. He always assumes perfect operation of the devices required in his designs.

The logic designer views the systems from an abstract angle. He builds his systems from simple logic blocks (NAND gates, inverters, etc.) and some standard complex functional units (multiplexers, registers and decoders). After designing such an unit, he is faced with the problem of testing it. He has little or no idea of the kinds of

functional faults that might arise from physical failures. This problem has been aggravated by rapid advances in process engineering and changes in the technology. He has, therefore, had to fall back on simple fault models which can be handled at the Boolean gate level. The simple stuck-line fault model is excellent for modeling physical failures in certain traditional technologies such as DTL and TTL. A great deal of work has, therefore, been done on methods to test logic units for stuck-at-1 and stuck-at-0 faults. Unfortunately, the development of fault models and logical models has not kept pace with VLSI technology. The inadequacy stems in part from the lack of formal logic models for describing the behavior of MOS circuits. Instead, systems are designed, simulated, and tested using an ad hoc combination of Boolean gate models, relay models, and electronic models.

This thesis has tried to bridge the gap between these areas for NMOS and CMOS technology, which are two of the most dominant technologies in VLSI today.

Information regarding typical physical failure modes observed in the field was obtained from the experts in that area - the process engineers. These were mapped into their effects at the circuit level. For example, a misalignment of a metal mask during fabrication might result in a resistive short between the gate and drain of a transistor in MOS technology; electro-migration can cause a high resistance, or an open in a line.

Next, typical circuit designs in both NMOS and CMOS technology of simple logic blocks (such as gates) and functional units were

considered. The effects of physical failures were then studied at the circuit level, using the SPICE simulation program. Results showed that an appreciable fraction of the faults was not covered by existing fault models.

A study of failures at the circuit level can provide accurate models for faults at the functional level. Such a study can also provide the logic designer with designs for improving the testability of certain functional blocks. Two such cases were discussed in the thesis.

The difficulties with this approach are that, firstly, it is implementation dependent, and secondly, it may be impossible to simulate large systems at the circuit level. The former is not so much of a problem since such a study might provide guidelines for improving the testability of a system. To overcome the latter problem, a multi-valued algebra was constructed to approximately model MOS circuit behavior under physical failures. The algebra was based on extensive studies of failures on small logic blocks.

With the help of the algebra, a new kind of MOS simulator was developed, which was more than 100 times faster than SPICE for simulating the same kinds of physical failures. The difficulty, however, was that no timing effects were modeled. Hence the failures that cause only timing errors but no logical errors cannot be correctly modeled in this simulator.

The difficulty with introducing accurate timing models in the form of an algebra, similar to the one discussed in the thesis, is that there too many variables involved. Introducing timing models in logic

simulators that can only simulate the behavior of fault-free MOS circuits is itself a rather difficult problem [10]. In addition, if we are to allow physical failures in those circuits, the problem becomes non-trivial. The state tables for the transistors would become so large that the memory requirement would get prohibitively expensive; so would the overhead in time involved in reading in such huge tables.

However, it may be possible to model the effects of physical failures on timing in some other elegant way. That is, perhaps, a topic for future research.

BIBLIOGRAPHY

[1] Aho, A. V., J. E. Hopcroft, and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison Wesley Publishing Company, 1974, pp. 187-195.

[2] Banerjee, P., and J. A. Abraham, "Fault Characterization of VLSI MOS Circuits," Proceedings of the IEEE International Conference on Circuits and Computers, New York, September 29 - October 1, 1982, pp. 564-568.

[3] Breuer, M. A., and A. D. Friedman, Diagnosis and Reliable Design of Digital Systems, Computer Science Press, Woodland Hills, California, 1976.

[4] Bryant, R. E., "An Algorithm for MOS Logic Simulation," Lambda Magazine, Fourth Quarter, 1980, pp. 46-53.

[5] Brzozowski, J. A., and M. Yoeli, "On a Ternary Model of Gate Networks," IEEE Trans. on Computers, vol. C-28, no. 3, March 1979, pp. 178-183.

[6] Carr, W. N., and J. P. Mize, MOS/LSI Design and Application, McGraw-Hill, New York, 1972.

[7] Case, G. R., and J. D. Stauffer, "SALOGS-IV: A Program to Perform Logic Simulation and Fault Diagnosis," Proceedings of the Fifteenth Design Automation Conference, IEEE New York, 1978, pp. 392-397.

[8] Chang, H. Y., et al., "LAMP: Logic Analyzer for Maintanence Planning," The Bell System Technical Journal, vol. 53, no. 8, October 1974, pp. 1431-1555.

[9] Chawla, B. R., H. K. Gummel, and P. Kozak, "MOTIS - An MOS Timing Simulator," IEEE Trans. on Circuits and Systems, vol. CAS-22, no. 12, December 1975, pp. 901-909.

[10] Easterbrook, J. T., and R. G. Benetts, "Failure Mechanisms in Logic Circuits and their Related Fault Effects," New Developments in Automatic Testing, Brighton, England, November 30 - December 2, 1977, IEEE London, pp. 44-47.

[11] Estriech, D. B., "Latch-up in CMOS Integrated Circuits," Defence Technical Information Center Research and Development Technical Report, no. AD A059752, July 1978, pp. 218-239.

[12] Farrow, R. H., and G. W. Parker, "The Failure Physics Approach to IC Reliability," Microelectron. Reliability, vol. 11, 1972, pp. 151-157.

[13] Harrison, M. A., Introduction to Switching and Automata Theory, McGraw Hill, New York, 1965.

[14] Hayes, J. P., "A Logic Design Theory for VLSI," Proceedings of the Second Caltech Conference on VLSI, Pasadena, California, January 1981.

[15] Hill, F. J., and G. R. Peterson, Introduction to Switching Theory and Logic Design, Wiley, New York, 1974.

[16] Hlavicka, J., and E. Kottek, "Fault Model for TTL Circuits," Digital Processes, Switzerland, vol. 2(3), Autumn 1976, pp. 169-180.

[17] Levendel, Y. L., P. R. Menon and C. E. Miller, "Accurate Logic Simulation Models for TTL Totempole and MOS Gates and Tristate Devices," The Bell System Technical Journal, vol. 60, no. 7, September 1981, pp. 1271-1287.

[18] Mak, G. P., J. A. Abraham and E. S. Davidson, "The Design of PLAs With Concurrent Error Detection," Proceedings of the Twelfth Fault Tolerant Computing Symposium, Los Angeles, CA, June 1982, pp. 303-310.

[19] Mead, C., and L. Conway, Introduction to VLSI Systems, Addison Wesley, Reading, Mass.(1979).

[20] Mei, K. C. J., "Bridging and Stuck-At Faults," IEEE Trans. Computers, vol. C-23, July 1974, pp. 720-726.

[21] Nagel, L., SPICE2: A Computer Program to Simulate Semiconductor Circuits, Technical Report UCB ERL-M250, Electronics Research Laboratory, University of California, Berkeley, May 1975.

[22] Newton, A. R., The Simulation of Large-Scale Integrated Circuits, Technical Report UCB ERL M78/52, Electronics Research Laboratory, University of California, Berkeley, July 1978.

[23] Rombeck, H., and P. Wilcox, "Interactive Logic Simulation and Test Pattern Development for Digital Circuitry," Electro 76, Paper 26.2, April 1976.

[24] Shannon, C. E., "A Symbolic Analysis of Relay and Switching Circuits," Trans. of the AIEE, vol. 57 (1938), pp. 713-723.

[25] Tanabe, N., H. Nakamura, and K. Kawakita, "MOSTAP: An MOS Circuit Simulator for LSI Circuits," Proceedings of the International Symposium on Circuits and Systems, IEEE Houston, Texas, April 1980.

[26] Thatte, S. M., and J. A. Abraham, "Test Generation for Microprocessors," IEEE Trans. Computers, vol. C-29, no. 6, June 1980, pp. 429-441.

[27] Utah, University of, VLSI Research Group, Simulog Manual, April 1979.

[28] Wadsack, R. L., "Fault Modeling and Logic Simulation of CMOS and NMOS Integrated Circuits," The Bell System Technical Journal, vol. 57, no. 2, May-June 1978, pp. 1449-74.

[29] Wadsack, R. L., "Fault Coverage in Digital Integrated Circuits," The Bell System Technical Journal, vol. 57, no. 5, May-June 1978, pp. 1475-1488.

[30] Yoeli, M., and S. Rinon, "Application of Ternary Algebra to the Study of Static Hazards," Journal of the ACM, ACM New York, vol. 11, no. 1, January 1964, pp. 84-97.

[31] Yuan, H. T., and S. M. Thatte, Texas Instruments Inc., Dallas, private communication.